The phfnote package[1]

Philippe Faist *philippe.faist@bluewin.ch*

October 08, 2021

[1] *This document corresponds to phfnote v4.0, dated 2021/10/08. It is part of the phfqitltx package suite, see https://github.com/phfaist/phfqitltx.*

phfnote—A handy LaTeX package for typesetting short notes and medium-length reports, full of goodies to make it look just right.

■ 1 Introduction

Have you ever thought, “let me write up these short notes using LaTeX,” but then disliked the default style of the `article` class? Have you ever asked yourself why half the page should be taken up by the title? Yes? Then welcome to phfnote.

The package phfnote provides basic formatting for short documents, such as notes on a specific topic, short documentation, or quick memos. It aims to cover all basic needs for such purposes: include a standard set of relevant packages, a nice title which doesn’t take up too much space, better page margin sizes, and some basic styling to make the note look nicer. At the same time, it is highly configurable so that nothing is really unchangeable. And all overridden features can be restored individually to their defaults provided by the underlying class.

This package has been designed to work optimally along with the `article` document class, but in principle any relatively standard LaTeX class should work. Notes can be typeset in two-column mode with the `twocolumn` option of for example the `article` class. Settings such as the page margins and font goodies are automatically adapted to look best according to the standard document font size (10pt, 11pt, or 12pt).

Be aware that this package is not meant as a full-fledged formatting class for complicated articles. For that, you should use a specialized class such as REVTeX.[1]

In the following, we detail individual features of this package, and explain how to activate, deactivate, and customize them.

2 Basic Usage

2.1 Loading the Package

You can get started with the minimal template:

```
\documentclass[11pt,a4paper]{article}
\usepackage{phfnote}

\begin{document}
\title{Title of my notes}
\author{Me}
\date{\today}
\maketitle

...

\end{document}
```

The package phfnote introduces its default note formatting style, with a more compact title, and some formatting adjustments in the text and section headings.

2.2 Presets

There are a number of package options which can be provided to activate, deactivate or adjust the formatting. The most straightforward way of changing the formatting is to use *presets*.

Presets are processed immediately when given in the package option list, meaning that their position in the list is meaningful. For example, the option list

```
\usepackage[title=small,preset=article,par=skip]{phfnote}
```

[1] See *https://journals.aps.org/revtex*

will set `title=small` only if it is not overridden by the `article` preset, but will enforce `par=skip` in any case. You may in theory load several presets, e.g. `preset=sfnote,preset=article`, but this is essentially useless since presets tend to set a wide range of settings such that in any case the last preset specified is effectively applied.

First, there is a set of presets which are different alternative "note" styles. All the following define the note to have spacing between paragraphs and no first line indentation, use the default note title style, and use a wider page geometry.

`preset=sfnote`

Format the note in LaTeX' sans-serif "Computer Modern Bright" font. This is a nice, light, font for short notes, but I find it more difficult to read at smaller font sizes or in longer paragraphs.

`preset=sfssnote`

Format the note in LaTeX' default sans-serif font. A very nice sans serif font. It might look heavy though, depending on your taste.

`preset=opensansnote`

Format the note in Open Sans font (using the 'opensans' package with some default options). A very beautiful and readable sans serif font.

`preset=utopianote`

Format the note in Utopia font (by using the fourier package). Perfect to my taste for documenting code for example, but I find it a bit heavy for scientific documents.

`preset=mnmynote`

Format the note in Minion Pro font, with sans serif text formatted with the Myriad Pro font (professional fonts by Adobe which can be used in LaTeX with the MinionPro and MyriadPro packages[2]). These beautiful fonts can be used for any purpose.

Based essentially on `utopianote`, the preset `pkgdoc` sets up the document to look nice for a LaTeX package documentation. The preset `xpkgdoc` adds additional definitions to aid in documenting LaTeX packages on top of `pkgdoc`.

`preset=pkgdoc`

Basic formatting and settings for documenting LaTeX packages. This preset was used for the current document.

`preset=xpkgdoc`

Same as `preset=pkgdoc`, but in addition a set of useful commands are

[2] See *https://github.com/sebschub/FontPro*; the fonts themselves ship with some Adobe products

also provided, the tcolorbox package is loaded along with some default boxes. Also some commands are patched to achieve some fixes. This preset is used for the documentation of packages in the phfqitltx package suite. (For details see the implementation of the `xpkgdoc` preset in subsubsection 5.14.1.)

The following preset makes the document look more like an article. There are some slight minor differences with respect to the default `article` class' title in the choice of formatting the title and text.

`preset=article`

Sets a more title style closer to `article`'s default title style (but slightly more compact) and sets paragraphs to indent with no skip.

The last preset, `reset`, guarantees that including this package is non-invasive, meaning that only new LaTeX macros are made available without altering any appearance. This is useful if you want to use a small feature provided by this package, but you already have all the page geometry, title, etc. set up and want to make sure those aren't touched.

`preset=reset`

Deactivates all features of this package by default. Individual settings can still later be switched on via specific package options. Use this to activate only a specific set of features: `[preset=reset,...]` will ensure that only the additional given features are set.

This is safer than deactivating individually all other features, because in the future we may add new features which may be on by default. In this case, the preset `reset` will guarantee all features to be deactivated.

3 Summary of Package Options

`preset=`⟨*preset name*⟩

Load a preset specifying a predefined set of options for the general appearance of the document. See documentation in subsection 2.2

`title=`⟨*title style*⟩ , `notitle`

Set the title style. Use `notitle` to disable feature and use latex default. Documentation in subsubsection 4.1.1

`abstract=`⟨*abstract attributes*⟩ , `noabstract`

Set the abstract style by specifying a comma-separated list of attributes. Don't forget to put the list of attributes within braces, `[abstract={wide,noname,it}]`. Documentation in subsection 4.2

`pkgset=`⟨*package set*⟩

Specify a standard set of LaTeX packages to load. See subsection 4.4.

`pagegeom=`⟨*geom style*⟩ , `nopagegeom`

Set a page margin style. Use `nopagegeom` to leave page geometry unchanged. Options are documented in subsection 4.5.

`secfmt=`⟨*section formatting attributes*⟩ , `nosecfmt`

A list of attributes defining how section (and possibly paragraph) headings should look like. See subsection 4.6.

`par=`⟨*par style*⟩ , `nopar`

Define how paragraphs should be spaced. Refer to subsection 4.7.

`spacingdefs` , `nospacingdefs`

Adjust spacing of lines and words (subsection 4.8).

`fontdefs` , `nofontdefs`

Adjust some fonts (subsection 4.9).

`footnotedefs` , `nofootnotedefs`

Adjust slightly the appearance of footnotes. See subsection 4.10.

`hyperrefdefs=`⟨*settings*⟩ , `nohyperrefdefs`

Load the hyperref package, and set some defaults settings. See subsection 4.11.

`bibliographydefs` , `nobibliographydefs`

Adjust the appearance and style of the bibliography. See subsection 4.12.

TIP

To activate only a subset of features, use `preset=reset` and then enable only the features required. In this way, you can ensure that only those features which are explicitly specified are enabled.

■ 4 Features

This package provides a large collection of small features, which, put all together, make the document look nicer (hopefully). Let's go through these features, one by one.

Note also that some features provided in the presets, such changing the document font, are not provided as individual features here. This is because they may be set and customized directly using few lines of LaTeX code or directly by including an external package. In those cases, you may have a look at the preset's definition for inspiration (see subsubsection 5.13.2).

For a summary of package options, see section 3.

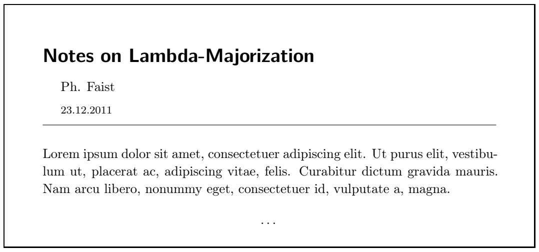
4.1 Title Formatting

4.1.1 Title Styles

The phfnote package allows a set of alternative title styles. By default, the `default` title style is used. You may change this setting with the `title=...` package option.

`title=default`

The default title style displays the title in large bold sans serif font, left-aligned. Below the title appears the information about author and date, indented, followed by a horizontal rule. It looks like this:



Notes on Lambda-Majorization

Ph. Faist

23.12.2011

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna.

...

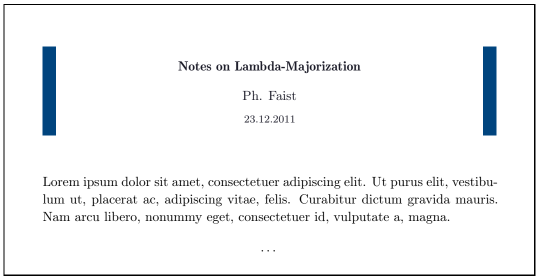
As you can see, it saves more space on the page compared to the default article title.

Changed in v3.0 [2018/10/25]: The default title style was redesigned to improved spacing of the elements in all cases including multiline titles, author and/or date not specified, and presence or absence of thanks notes. Use `title=defaultv1` for old behavior.

If you would like your document to appear exactly as with phfnote version 1.0, then you can set the special title style `title=defaultv1`.

`title=pretty`, `title=pretty2`

A prettier, fancier title style. The title is centered, with two side bars providing a visual guide. The `pretty` style looks like this:



Notes on Lambda-Majorization

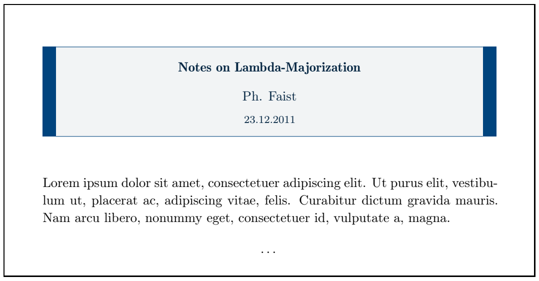
Ph. Faist

23.12.2011

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna.

...

And the `pretty2` style looks like this:



Notes on Lambda-Majorization

Ph. Faist

23.12.2011

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna.

...

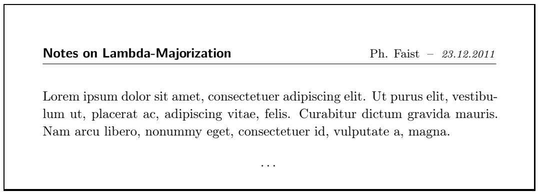
Changed in v3.0 [2018/10/25]: Added the `pretty` and `pretty2` title styles.

This title style is highly customizable (see below), in fact the `pretty2` style is an alias for the `pretty` style, with adjusted settings.

These styles require the `xcolor` package (it will be loaded automatically).

`title=small`

A smaller title style which displays all the relevant information on a single line. This is useful for when even the default title style appears too large. It looks like this:



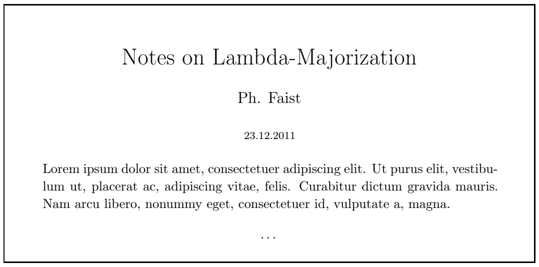
Notes on Lambda-Majorization Ph. Faist – *23.12.2011*

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna.

...

`title=article`

Mimics the default title style from the `article` class, but saves a little more space. It looks like this:



Notes on Lambda-Majorization

Ph. Faist

23.12.2011

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna.

...

Changed in v3.0 [2018/10/26]: The `article` title style was redesigned to use our new title engine, with improved spacing of the elements in all cases including multiline titles, author and/or date not specified, and presence or absence of thanks notes. Use `title=articlev1` for old behavior.

If you would like your document to appear exactly as with phfnote version 1.0, then you can set the special title style `title=articlev1`.

`notitle`

Also equivalently, `title=false`. Instructs phfnote not to override any title definition, thus preserving the default class title style.

Beware that some other title goodies, such as our more advanced `\thanks` notes, or spacing adjustments for the abstract, will probably not work.

For compatibility with previous versions of `phfnote`, you may also specify an empty option value "`title=`".

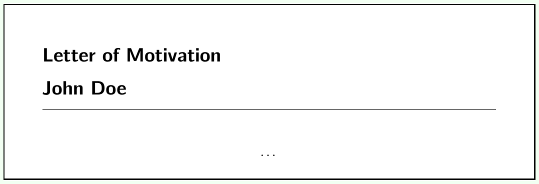
TIP

When using the `default` and `pretty*` title styles, the argument to `\title` may contain blank lines. In this case, each part is typeset on a separate line with an appropriate spacing. For instance:

```
\title{Letter of Motivation

John Doe}
```

will be typeset as



Letter of Motivation

John Doe

...

Changed in v3.0 [2018/10/16]: Improved `title,notitle` package options syntax.

4.1.2 Customizing the style of the `default`, `pretty` and `small` title styles

You may customize the appearance of the `default` and `small` title styles by overriding some macros.

`\notetitlefont`
`\notetitleauthorfont`
`\notetitledatefont`

The macros `\notetitlefont`, `\notetitleauthorfont`, and `\notetitledatefont` set the default main font title, author text and date text. You may override these settings with, for instance:

```
\renewcommand{\notetitlefont}[1]{\sffamily\bfseries #1}
```

```
\renewcommand{\notetitleauthorfont}[1]{\itshape #1}
\renewcommand{\notetitledatefont}[1]{\footnotesize #1}
```

In these macros, the parameter is expanded to the value provided by `\title`, `\author`, and `\date`, respectively.[3]

`\notetitlebelowspace`
`\notetitletopspace`

The spacing of the title may be adjusted with the macros `\notetitlebelowspace` and `\notetitletopspace`. Override these with e.g.:

```
\renewcommand{\notetitlebelowspace}{4mm}
\renewcommand{\notetitletopspace}{-1.2cm}
```

`\notetitlehrule`

Finally, you may override the command `\notetitlehrule` which draws the rule below the title:

```
\renewcommand{\notetitlehrule}{\hrule height 0.8pt}
```

`\notetitleusempfootnotestrue`
`\notetitleusempfootnotesfalse`

The commands `\notetitleusempfootnotestrue` and `\notetitleusempfootnotesfalse` set respectively whether any `\thanks` commands in the title generate footnotes inside the title area (which is drawn within a minipage), i.e., all thanks notes are collected on an additional line below the date, or whether they appear as regular footnotes at the bottom of the page. Simply call the relevant command to set either setting (don't redefine these).

The `small` title style allows you to customize the separator between the author and the date:

```
\renewcommand\notetitlesmallauthordatesep{\hspace*{0.5em}\cdot\hspace*{0.5em}}
```

In addition to the above settings for the default title style, the `pretty` and `pretty2` styles provide a few macros that you can adjust some visual aspects to your needs:

```
\renewcommand\notetitleprettylsiderulewidth{10pt}
\renewcommand\notetitleprettylsidespacewidth{10pt}
\renewcommand\notetitleprettyrsiderulewidth{10pt}
\renewcommand\notetitleprettyrsidespacewidth{10pt}
\renewcommand\notetitleprettytopspace{10pt}
\renewcommand\notetitleprettybottomspace{10pt}
\renewcommand\notetitleprettytophrulewidth{0pt}
\renewcommand\notetitleprettybottomhrulewidth{0pt}

\colorlet{notetitleprettylsiderulecolor}{blue!40!black}
```

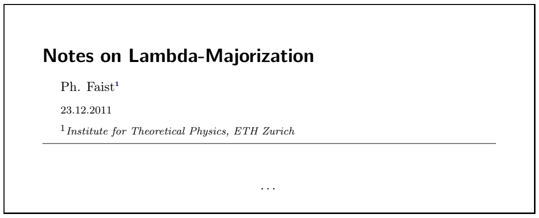
[3] While you can normally define \notetitlefont, \notetitleauthorfont and \notetitledatefont without an explicit parameter, this might produce some unexpected errors in some cases since some title styles (for technical details see implementation of the `article` title style).

```
\colorlet{notetitleprettyrsiderulecolor}{notetitleprettylsiderulecolor}
\colorlet{notetitleprettytophrulecolor}{notetitleprettylsiderulecolor}
\colorlet{notetitleprettybottomhrulecolor}{notetitleprettylsiderulecolor}
\colorlet{notetitleprettytextcolor}{blue!20!black}
\colorlet{notetitleprettybgcolor}{white!95!blue}
```

4.1.3 Title notes: `\thanks` and `\thanksmark`

`\thanks` Notes in the title can be introduced with the `\thanks` macro. You may use this to specify an e-mail address, an affiliation, or any other more specific information. `\thanks` may appear in all three title, authors and date.

The appearance of this additional information depends on the title style. In the default note title style, such thanks-notes appear directly below the title. For example, with `\author{Ph. Faist\thanks{\itshape Institute for Theoretical Physics, ETH Zurich}}`, you get:



Notes on Lambda-Majorization

Ph. Faist[1]

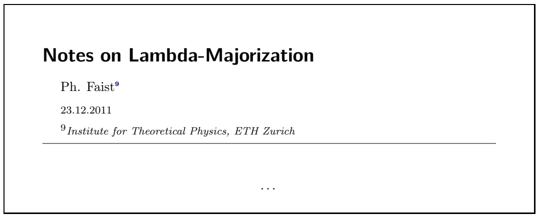
23.12.2011

[1] *Institute for Theoretical Physics, ETH Zurich*

...

whereas with the other styles, this information is typeset as regular footnotes.

`\thanks[N]` You may specify an optional argument to `\thanks`, forcing the footnote to a specific number (it must be a number). For example, with `\author{Ph. Faist\thanks[9]{\itshape Institute for Theoretical Physics, ETH Zurich}}`, you get:



Notes on Lambda-Majorization

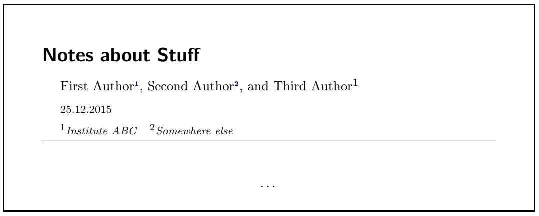
Ph. Faist[9]

23.12.2011

[9] *Institute for Theoretical Physics, ETH Zurich*

...

`\thanksmark` `\thanksmark[N]` works with `\thanks` as `\footnotemark` works with `\footnote`. It just displays the given number as a footnote mark. In this way, you can have for example several shared affiliations:



Notes about Stuff

First Author[1], Second Author[2], and Third Author[1]

25.12.2015

[1] *Institute ABC* [2] *Somewhere else*

. . .

the author code was:

```
\author{First Author\thanks[1]{\itshape Institute ABC},
    Second Author\thanks[2]{\itshape Somewhere else},
    and Third Author\thanksmark[1]}
```

Unfortunately, you still have to provide the numbering manually. On the other hand, this package is not meant to replace REVTEX, so if you're writing a complicated article with many authors and affiliations, you probably shouldn't be using phfnote in the first place.

WARNING

The optional argument to `\thanks`, as well as the command `\thanksmark`, are not made available if you don't use one of phfnote's title styles.
This behavior is such as to prevent interference with more advanced class mechanisms, such as REVTEX's.

TIP

For phfnote's title styles, you can issue the commands `\notetitleusempfootnotestrue` or `\notetitleusempfootnotesfalse` (documentation in subsubsection 4.1.2) to decide whether the thanks notes are issued in a separate title line, or if they are displayed as regular footnotes at the bottom of the page.

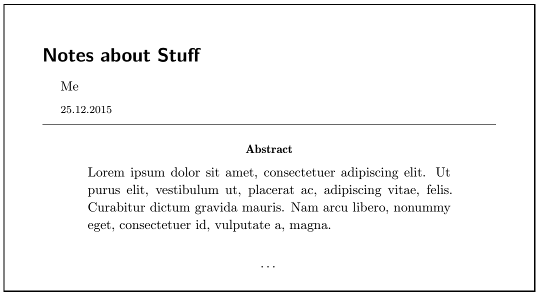
4.2 Abstract

`abstract` The `abstract` environment renders indented text aimed to provide a short summary of the document. We might use, for example, the following code:

```
\begin{abstract}
Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus
elit, vestibulum ut, placerat ac, adipiscing vitae, felis.
Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget,
consectetuer id, vulputate a, magna.
```

```
\end{abstract}
```

which would look like this:



Notes about Stuff

Me

25.12.2015

Abstract

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna.

...

The `abstract` environment should be given *after* the `\maketitle` command. (In contrast to, e.g., RevTeX.)

You may customize the appearance of the abstract via a list of attributes given as argument to a package option. When you combine arguments, make sure to put them in a braced group: `[abstract={wide,noname,it}]`.

`abstract={wide,...}`

The abstract should not be indented, and should instead be aligned to the rest of the text.

`abstract={narrow,...}`

The abstract should be indented narrower then by default.

`abstract={noname,...}`

The title “Abstract.” above the text will not be typeset. The abstract text is typeset directly instead.

`abstract={small,...}`

Use a smaller font for the abstract text (`\small` font).

`abstract={compact,...}`

Reduce spacing before and after the abstract. If the abstract is short, this might look slightly better.

`abstract={it,...}`

Typeset the abstract text using an italic typeface.

`noabstract`

Do not (re)define the `abstract` environment, do not execute abstract definitions. This leaves the original `abstract` environment definition of the underlying LaTeX class. You can also equivalently say `abstract=false`.

`\noteabstracttextfont` `\noteabstractnamefont` `\noteabstracttextwidth` `\noteabstractafterspacing` `\noteabstractbeforespacing`

The abstract environment's appearance can be customized more finely by redefining some macros. (In fact, this is what the package options `abstract=...` actually do.) The font used for the text of the abstract is set by `\noteabstracttextfont`. This macro should expand to font selection commands, such as `\itshape`, `\bfseries`, `\small`, etc. The title of the abstract (the word "Abstract.") is typeset in the font set by `\noteabstractnamefont`. The width of the whole abstract text is determined by `\noteabstracttextwidth`. Observe that `\noteabstracttextwidth` is a macro, and not a proper length, so that it can determine more dynamically the length. The spacing below (`\noteabstractafterspacing`) and above (`\noteabstractbeforespacing`) the abstract can further be specified, also as macros.

Obsolete options:

`abstract={original,...}`

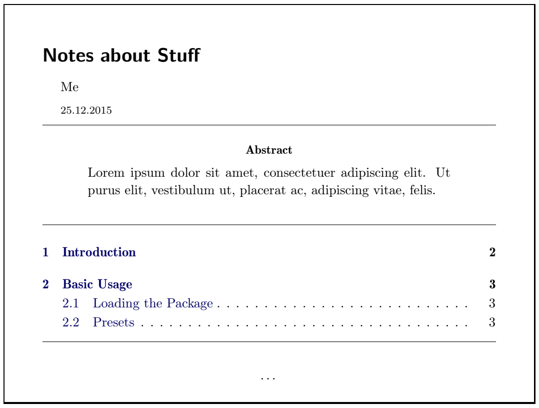
Revert to the original class' default implementation of the `abstract` environment before phfnote was loaded. The original class' implementation is restored and no longer tampered with. This option is OBSOLETE, use `noabstract` instead.

Changed in v3.0 [2018/10/16]: Improved `abstract,noabstract` package options syntax.

4.3 Table of Contents

`\inlinetoc`

The package phfnote also provides a table of contents typeset with reduced spacing to be more compact, and with horizontal rules before and after. You can insert the table of contents with the command `\inlinetoc`. It looks like this:



Notes about Stuff

Me

25.12.2015

Abstract

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis.

1 Introduction **2**

2 Basic Usage **3**

2.1 Loading the Package 3

2.2 Presets 3

. . .

4.4 Predefined Package Sets

The phfnote package also provides sets of standard LaTeX packages to load. You may choose between a varying degree of “richness” of packages included.

`pkgset=none`

Do not include any package set.

`pkgset=minimal`

Include some basic minimal set useful for scientific notes: the $\mathcal{AMS}$ packages amsmath, amssymb, amsfonts, and amsthm. The xcolor package is also loaded.

`pkgset=rich`

Include a fair amount of packages which may be useful. On top of the `minimal` package set, this set includes the packages enumitem, graphicx, microtype, caption, setspace, as well as inputenc with the `utf8` option and fontenc with the `T1` option (the packages fontenc and inputenc are not loaded if running Xe(La)TeX or Lua(La)TeX, or if they are already loaded with possibly different options).

This package set is loaded by default.

Changed in v1.1 [2018/08/27]: If running XeTeX or LuaTeX, then do not load inputenc and fontenc as part of `rich` and `extended` package sets. Plus, do not load inputenc (resp. fontenc) if the package is already loaded.

`pkgset=extended`

Additionally, include packages float, verbdef, csquotes, dsfont, bbm and mathtools.

4.5 Page Geometry

Another important aspect of phfnote is the handling of page margins. Often the default page margins of the article class are quite narrow. While it is a good typographical practice to avoid long lines, on occasion we prefer to have notes typeset with wider text. The general answer is the geometry package, which allows to set all margins in full detail.

The phfnote package provides some standard choices of options for the geometry package, which are adjusted according to the document font size, and whether the document is typeset in two columns.

If you want anything more complicated than what is provided by a default setting here, just use the `nopagegeom` package option and invoke the geometry package directly with your preferred set of options.

The page geometry predefined settings are the following.

`pagegeom=default`

Default settings. Not too wide, not too narrow. Settings vary according to single or double column setting, and according to default font point size.

`nopagegeom`

Also, `pagegeom=false`. Do not change page geometry settings, do not load the geometry package.

`pagegeom=narrow`

Narrower style. For single-column documents, this is closer to the typographically-advertised-optimal of 50–80 characters per line, but it might look narrow to some.

`pagegeom=wide`

Wide, comfortable style. Wastes less paper.

`pagegeom=xwide`

Extra wide. Use if you pity trees.

`pagegeom=bigmargin`

Makes the margins asymmetric, so that a wide margin note can fit. This style is used in this package documentation, for example.

The following package options are OBSOLETE:

`pagegeomdefs=`⟨*true or false*⟩

Whether to care about page margins. `nopagegeomdefs` is synonym for `pagegeomdefs=false`. This option is OBSOLETE, please use `nopagegeom` or `pagegeom=false` instead of `pagegeomdefs=false` and `pagegeom` (or `pagegeom=...` with setting) instead of `pagegeomdefs=true`.

Changed in v3.0 [2018/10/16]: Changed how to turn on/off the page geometry settings by improving the `pagegeom` package options, deprecated `pagegeomdefs` option.

Changed in v3.0 [2018/11/30]: Changed the `xwide` page geometry for tighter vertical margin for single-column text (use `pagegeom=xwidev1` instead for old behavior).

4.6 Section Headers Styling

The phfnote package provides some limited styling of section headers. The font, size and “compactness” of the headers can be adjusted with title options. But really, these options are quite basic. You should use titlesec or sectsty directly if you want anything serious.

The section headings are customized using the sectsty package. If this conflicts in your document, then use the `nosecfmt` package option to indicate that section headings should NOT be styled by this package. Then you have the full freedom to take care of section styling manually.

Package options may be used to customize the appearance of the section headings by specifying a list of attributes. When you combine arguments, make sure to put them in a braced group: `[secfmt={section,compact}]`. Beware that attributes are not merged between different occurrences of the `secfmt` keyword in the package options; the last occurrence defines all set attributes. If the `secfmt` package option is not given, then by default only the `section` attribute is set.

NOTE

Don't forget to include the attribute '`section`' and/or '`paragraph`' depending on which type of heading you want your settings to apply to. For example, `secfmt={sffamily}` has no effect, you need to use e.g. `secfmt={section,sffamily}`.

Available attributes are the following:

`secfmt={section,...}`

Use the `section` attribute to activate the styling of section-level headings, that is, `\section`, `\subsection` and `\subsubsection`.

`secfmt={paragraph,...}`

This attribute indicates that the styling should apply to paragraph-level headings as well (`\paragraph` and `\subparagraph`).

`secfmt={compact,...}`

Reduce the sizes of the section headings (if the section-level headings are styled, i.e. you need to specify the `section` attribute), giving the document a more "compact" appearance.

`secfmt={larger,...}`

Increase the sizes of the section headings. Suitable for longer documents or for small document font sizes.

`secfmt={secsquares,...}`

Display black squares on the left side of `\section`-level commands, making them stand out better. This is useful for documents (such as the present one) with several layers of sub-sections.

`secfmt={secnummargin,...}`

Display the section, subsection, and subsubsection numbering in the left margin and have the title occupy the full width of the text (such as for this document). If you want both `secsquares` and `secnummargin`, you must specify them in that order, or the black square may end up overlapping with the number.

`secfmt={rmfamily,...}`

Typeset headings in the regular roman font of the document, instead of trying to apply the **Palatino font**. This applies to section-level and/or paragraph-level headings, depending on which of the attributes `section` and/or `paragraph` have been specified.

`secfmt={sffamily,...}`

Typeset headings in a sans-serif font. The default document sans serif font is used. This applies to section-level and/or paragraph-level headings, depending on which of the attributes `section` and/or `paragraph` have been specified.

`secfmt={itpar,...}`

Typeset paragraph-level headings in italic.

`secfmt={blockpar,...}`

Change the paragraph-level headings not to be in “run-in” style, but to be typeset on their own line like section headings.

`nosecfmt`

Keep the original class styling; nothing will be overridden and the sectsty package is not loaded. (Equivalently, you may specify `secfmt=false`.)

You can also directly modify the section heading style by redefining some macros. Note that these macros only affect those sectioning commands which we have decided to style, which is specified by the `section` and `paragraph` attributes to be specified in the `secfmt={...}` package option.

`\notesectionallfont`

`\notesectionallfontfamily`

The macro `\notesectionallfont` is invoked for every sectioning command (for those which are styled, see the `section` and `paragraph` attributes). The macro `\notesectionallfont` internally invokes `\notesectionallfontfamily` to select which font family to use. The family should be given as the font code, e.g.: `pbk` = Bookman; `bch` = Charter; `ppl` = Palatino; `ptm` = Adobe Times; `phv` = Adobe Helvetica; `pcr` = Adobe Courier; `put` = Utopia; `cmr` = Computer Modern Roman; `cmss` = CM Sans Serif; `cmbr` = CM Bright; google many more or look directly into the source of corresponding LaTeX packages.

You may customize these either via attributes or by redefining them directly. Beware that if you redefine `\notesectionallfont` then you are responsible for honoring, or ignoring, the value of `\notesectionallfontfamily`.

`\notesectionfont`
`\notesubsectionfont`
`\notesubsubsectionfont`
`\noteparagraphfont`
`\notesubparagraphfont`
`\notesectionsetfonts`

These macros define the font commands to apply for the section heading corresponding to the given sectioning command. This macro is invoked after `\notesectionallfont`, which means that font definitions in these macros take precedence over those in `\notesectionallfont`.

The macro `\notesectionsetfonts` is a shorthand to set all section font definitions for the section-level commands `\section`, `\subsection`, and `\subsubsection`. For example,

```
\notesectionsetfonts{\Large}{\large}{\normalsize}
```

will set the font sizes for `\section`, `\subsection` and `\subsubsection` in this order.

`\noteparagraphsetfonts`

The macro `\noteparagraphsetfonts` is the corresponding shorthand for the paragraph-level commands. It takes two arguments, the font definitions to apply for headings of level `\paragraph` and `\subparagraph`.

Obsolete options:

`secfmt={}`

> Leave the argument empty to keep the original class styling; nothing will be overridden and the sectsty package is not loaded. This option is OBSOLETE, use `nosecfmt` instead.

Changed in v3.0 [2018/10/16]: Improved `secfmt,nosecfmt` package options syntax.

4.7 Appearance of Paragraphs

Several presets may be set to define the appearance of paragraphs.

`par=indent`

> Paragraphs are indented, bearing some similarity to the article class' default paragraph style.

`par=skip`

> Paragraphs are separated by additional spacing, and not indented.

`par=indentminiskip`

> Paragraphs are indented, but there is also a small space between each paragraph.

`nopar`

> Do not modify the appearance of paragraphs, and leave the original class' default.
>
> You may also use `par=false`.

Obsolete options:

`par=original`

OBSOLETE—use `nopar` instead.

Changed in v3.0 [2018/10/16]: Improved `par,nopar` package options syntax.

4.8 Adjusting Spacing of Lines and Words

The phfnote package also provides definitions to adjust spacing of lines and words.

This includes definitions to avoid overflowing words in the margin in case of long words.

`spacingdefs`

Apply adjustments to line and word spacing.

This feature is on by default. You can also use `spacingdefs=true`.

`nospacingdefs`

Do not attempt any adjustments of line or word spacing. You can also use the alias `spacingdefs=false`.

4.9 Adjustments for Fonts

The phfnote package provides as well some adjustments for fonts to make some fonts look nicer.

Concretely, the Computer Modern Bright font is used as sans serif font instead of LaTeX's default sans serif font, and the more universal `T1` font encoding is used instead of the default `OT1`.

`fontdefs`

Apply adjustments to fonts. This is on by default, except on XeTeX and LuaTeX.

You can also use `fontdefs=true`.

`nofontdefs`

Do not apply adjustments to fonts. You can also set `fontdefs=false`.

4.10 Footnote Style

The footnotes' appearance can also be slightly enhanced.

`footnotedefs`

Changes the symbol appearance a little bit—the footnote number is smaller and typeset in boldface.

You can also use `footnotedefs=true`. This feature is on by default.

`nofootnotedefs`

Do not change the footnote appearance. You can also set `footnotedefs=false`.

4.11 Hyperref Loading

There are many options for setting up the hyperref package, and often, the defaults (with boxed links) are pretty ugly in my opinion. Enable the `hyperrefdefs` feature of phfnote to alter the defaults to something I personally like better (dark blue links as in this document).

`hyperrefdefs`

Load the hyperref package, and set some sensible default settings. Also ensures the `\email` and `\url` commands are made available.

You can also use `hyperrefdefs=true`. This feature is on by default.

`nohyperrefdefs`

Do not load the hyperref package. Do not set any settings. Do not care to provide `\email` or `\url`. Same as `hyperrefdefs=false`.

Depending on the situation, you might prefer to specify `hyperrefdefs=defer` or `hyperrefdefs=noload`, so that some basic setup (e.g. `\url`/`\email` commands) can still be provided. See below.

`hyperrefdefs={defer,...}`

Prepare the document for hyperlinks, schedule settings for hyperref, but do not actually load the hyperref package.

This is useful if you would like to load more packages that need to be loaded before loading hyperref. A lot of packages need to be loaded before hyperref so if you load several other packages, you're probably better off using this option and calling `\usepackage{hyperref}` at the end of your preamble, i.e. right before `\begin{document}`, rather than chasing mysterious errors.

When using this option it is the user's responsibility to load the load the package with `\usepackage{hyperref}` somewhere in the preamble. You will get an error if you don't do this.

`hyperrefdefs={noemail,...}`

Do not override any existing `\email` command. Use this for instance in RevTEX, where our implementation of `\email` clashes with RevTEX's `\email` command which is used to specify e-mail addresses for authors.

The version of this command by phfnote is still available as `\phfnoteEmail`.

`hyperrefdefs={noeqref,...}`

Do not redefine the `\eqref` command to include the parenthesis in the hyperlink.

Changed in v3.2 [2021/07/29]: We now redefine `\eqref` by default to include the parentheses inside the hyperlink. Use the `hyperrefdefs=noeqref` package option to disable this feature.

`hyperrefdefs={noload,...}`

Do not load the hyperref package, and don't bother to set any related settings. However, url package is loaded, and the commands `\url` and `\email` are provided (they output the same visual text but don't produce a clickable color link).

`hyperrefdefs={clearoptions,...}`

Do not attempt to set any options via `\hypersetup` (and don't schedule setting any such options later). You'll get hyperref's default settings, so it's up to you to call `\hypersetup` with however you'd like to see your links look like.

Attributes may be combined, as in `hyperrefdefs={noemail,noload}`. In this case make sure you put them in a braced group. Also, beware that attributes are not merged between different occurrences of the `hyperrefdefs` keyword in the package options; the last occurrence defines all set attributes.

When the `hyperref` package is loaded, it is done so with the `unicode=true` package option. In case you need, you can specify your own package options with `hyperrefdefs=defer` and then calling `\usepackage[...]{hyperref}`. For most options though it's simpler to use `\hypersetup{...}`.

`\url` In order to typeset URLs, the `\url` command is made available from the package url (which is then linkified by hyperref). For example, you can type `\url{https://github.com/phfaist/}`.

`\email`
`\phfnoteEmail`
A similar command allows to typeset e-mail addresses. The text is displayed as a hyperlink, which when clicked opens a e-mail composer to that address (via a `mailto:XXX` link). For example, try `\email{pulp_fiction@tarantino.com}`.

The command \phfnoteEmail is an alias for this, which is defined even if the noemail attribute is given.

The commands \url and \email (along with \phfnoteEmail) are defined unless nohyperrefs (or hyperrefdefs=false) is specified. If you would like to use these commands but not load the hyperref package, consider using hyperrefdefs=noload.

\phfnotePdfLinkColor

The command \phfnotePdfLinkColor may by used to set the color of the links. It takes one argument, a color specification understood by the xcolor package. For example:

```
\phfnotePdfLinkColor{green!50!black}
```

> **NOTE**
>
> The package xcolor must be loaded for \phfnotePdfLinkColor to work. (The xcolor package is automatically loaded as part of a package set as long as you're not using the option pkgset=none; see subsection 4.4.)

The internal name for the link color is docnotelinkcolor. (This name is historical, and I'm not really willing to change it.)

4.12 Bibliography Definitions

This package also provides some definitions for the bibliography.

It sets the naturemagdoi style by default, which is a hacked (by yours truly) version of the naturemag style to include the journal name as a hyperlink (as in APS bibliography styles).

The bibliography is also typeset in a smaller font.

Finally, an entry in the table of contents is generated.

`bibliographydefs`

Set some default bibliography settings.

This feature is on by default. You can also use bibliographydefs=true.

`nobibliographydefs`

Do not set some bibliography settings. You may also use bibliographydefs=false.

\bibliography
\bibliographystyle

The \bibliographystyle and \bibliography macros can be used as usual, for example:

```
\bibliographystyle{apsrmp4-1} % optional
\bibliography{mybibfile}
```

bearing in mind that if the `\bibliographystyle` command is not present, our custom `naturemagdoi` bibliography style is used.

4.13 Inline Commenting in Documents

Inline commenting features (`\phfMakeCommentingCommand`) have been moved to the separate dedicated package phfcc. If you were using these features in your document, simply do

```
\usepackage{phfcc}
```

and everything should work as expected.

Changed in v3.0 [2018/10/03]: Added support for inline commenting using `\phfMakeCommentingCommand`.

Changed in v3.1 [2020/04/02]: Moved support for inline commenting to the separate dedicated package phfcc.

4.14 URL Styles

As a bonus, the phfnote package provides an alternative set of URL styles to use with the `\url` and `\email` commands (see subsection 4.11).

All the styles described below typeset the URL in a slightly smaller size, so as to avoid a common issue with URLs that they tend to appear too large. Also, the tilde character is fixed so that it appears nicely, as in: *https://people.phys.ethz.ch/~pfaist/*.

The URL style can be set with the command `\urlstyle{`⟨*name of style*⟩`}`.

`notett`	typewriter font
`notesf`	default sans serif font
`notesfss`	Computer Modern Sans Serif font
`noteitsf`	italic using default sans serif font
`noterm`	normal roman typeface
`noteit`	just italic typeface
`notesml`	just smaller than surrounding text

4.15 A `\notesmaller` Command

This general-purpose command is handy to typeset text smaller than its surrounding text, for when you don't know what size the surrounding text is typeset at. In some sense, this is a very very lightweight analogue of what the relsize package does. (This is used, for example, in our implementation of URL styles introduced in subsection 4.14.)

`\notesmaller`
`\notesmaller[0.8]`

Set the font size to a fraction of the surrounding font size. The fraction may be specified as an optional argument. A fraction of 0.8 makes the text size 0.8 times that of the surrounding text, that is, smaller than the surrounding text. A value of 1 does not change the font size. If the fraction is not specified, the value stored in `\notesmallerfrac` is used.

`\notesmallerfrac`

The fraction by which `\notesmaller` typesets smaller text when no optional argument is given. You may redefine this command to set the default "smaller" size fraction.

4.16 Tools Mostly for Hackers

`\phfnoteHackSectionStarWithTOC`
`\phfnoteSaveDefs`
`\phfnoteRestoreDefs`

The phfnote package also provides some small hacks. They are documented further in subsection 5.12. These are: a macro `\phfnoteHackSectionStarWithTOC` to hack into a command which generates a `\section*`, in order for that command to also generate a corresponding entry in the table of contents; and a pair of commands to save and restore LaTeX definitions.

5 Implementation

Here comes the gory code.

Let's start by loading the kvoptions package, which we need to parse the package options. It's better to use xkeyval as backend, because the `\setkeys` by keyval is a little fragile: for example, it gets confused if, within a preset, we include a package or run a command which itself parses key-vals.

```
\RequirePackage{xkeyval}
\RequirePackage{kvoptions}
```

Also load etoolbox, for various utilities and xparse (for parsing optional arguments with recursive matching open/close brackets).

```
\RequirePackage{etoolbox}
\RequirePackage{xparse}
```

5.1 Internal Generic Code

\phfnote@internal@execattribs

An internal general-purpose macro to execute all definitions given in list of attributes.

Often, a list of attributes are given via a package option (e.g. for the abstract), and these attributes need to be executed, or implemented, in the order they are given. This macro takes care of that. Each possible attribute must be defined as a macro with a common prefix, to which the attribute is appended.

The arguments are:

- #1 = prefix to look for attributes (e.g. `noteabstract@attr@`);
- #2 = a human-readable name of what #1 represents, which is used in an error message in case the required attribute is not found (e.g. `{abstract attribute}`);
- #3 = the list of attributes specified by the user.

For example, `\phfnote@internal@execattribs{noteabstract@attr@}{abstract attribute}{noname,small}` causes the commands `\noteabstract@attr@noname` and `\noteabstract@attr@small` to be invoked, in this order.

```
\def\phfnote@internal@execattribs#1#2#3{%
  \@for\next:=#3\do{%
    \ifcsname #1\next\endcsname%
      \csname #1\next\endcsname%
    \else%
      \PackageError{phfnote}{Unknown #2: ’\next’. Ignoring.}{The given #2 ’\next’
        is invalid.  Consult the package documentation for information about
        valid attributes.}
    \fi
  }
}
```

5.2 Title Styling

See subsubsection 4.1.1 for a description of the styles and which features are available.

5.2.1 First, some common simple definitions for our different styles

\notetitlefont
\notetitleauthorfont
\notetitledatefont

These may be redefined to adapt the font of the title, author and date.

```
\newcommand{\notetitlefont}[1]{\sffamily\bfseries #1}
\newcommand{\notetitleauthorfont}[1]{#1}
\newcommand{\notetitledatefont}[1]{\footnotesize #1}
```

\notetitlebelowspace
\notetitletopspace

These macros may be redefined to adjust spacing above and after the title. They are macros, not lengths, so they can be adjusted dynamically on the spot.

```
\newcommand{\notetitlebelowspace}{4mm}
\newcommand{\notetitletopspace}{-1.2cm}
```

\notetitlehrule

Allow customization of the horizontal rule below the title. The macro `\notetitlehrule` expands to commands which generate the rule, such as "`\hrule height 1pt`".

```
\newcommand{\notetitlehrule}{\hrule}
```

\notetitle@title

Provide a "long" definition for `\title`, so that the title can have several paragraphs. Our style handles this by putting the title on several lines, and it can be useful depending on how you want to format the title.

This macro will replace `\title` when a title style is actually selected in `\phfnote@do@notetitle`.

```
\long\def\notetitle@title#1{\long\gdef\@title{#1}}
```

5.2.2 Implementation of `\thanks` and `\thanksmark`

Here we provide a few fixes for the implementation of `\thanks`, both for our main 'default' title style as well as for other simpler styles. Our implementation supports `\thanks[N]{...}` and `\thanksmark[N]` as for footnotes.

These newer implementations are only applied if one of our title styles is set. Otherwise, the class defaults are left untouched (which may be needed, e.g., for RevTeX).

Implementation of `\thanks` *and friends for our main 'default' title style*

\phfnote@setupthanksmpfootnote

Internal—called at the beginning of a `minipage` environment, it sets up necessary stuff to support `\thanks` notes within the minipage, in a single paragraph.

Some of this code was taken or really inspired directly from `latex.ltx`.

```
\def\phfnote@setupthanksmpfootnote{%
```

The \thanks macro is implemented as a \footnote in a minipage. So we hack into the 'mpfootnote' mechanism.

```
  \def\thempfootnote{\arabic{mpfootnote}}%
  \let\footnoterule\relax%
  \let\thanks\footnote%
```

All footnote material is stored in a macro \phfnote@mpfootmaterial, initially empty:[4]

```
  \def\phfnote@mpfootmaterial{}%
```

and locally define \@mpfootnotetext to store the footnote content into that buffer,

```
  \long\def\@mpfootnotetext##1{%
    \protected@edef\@currentlabel%
         {\csname p@mpfootnote\endcsname\@thefnmark}%
    \protected@edef\@tmpa{\protect\phfnote@mympfootnotemark{\@thefnmark}{##1}%
      \protect\phfnote@mpfootnoteglue}%
    \expandafter\g@addto@macro\expandafter\phfnote@mpfootmaterial%
      \expandafter{\@tmpa}%
  }%
```

Also provide \thanksmark, so that we can refer to other thanks/footnote-marks.

```
  \def\thanksmark[##1]{\phfnote@mympfootnotemark{##1}}%
}
```

\phfnote@finalizempfootnotes

Macro to call at the end of a minipage environment, to ensure that all \footnote's (and thus \thanks's) are properly formatted.

This simply takes all the tokens collected in \phfnote@mpfootmaterial (see just above), and typesets it in the \@mpfootins box. The latter is automatically typeset by the minipage in \end{minipage}.

The argument #1 is the skip length between the text and the footnotes.

```
\def\phfnote@finalizempfootnotes#1{%
  \if\relax\detokenize\expandafter{\phfnote@mpfootmaterial}\relax
  \else
    \global\skip\@mpfootins=#1\relax
    \global\setbox\@mpfootins=\vbox{%
      \parskip=\z@\relax
      \parindent=\z@\relax
      \phfnote@mpfootnotes@fontparsetup
      \noindent\leavevmode%
      \reset@font\footnotesize%
      \phfnote@fmt@titlefootnotes%
```

[4] NOTE: this differs from how footnotes are usually treated (directly typeset into a vbox I think). Not sure what the side-effects might be. Because this is just for simple email/institute info/etc. in the title, hopefully this shouldn't have any serious consequences.

```
      \phfnote@mpfootmaterial}%
  \fi
}
```

\phfnote@fmt@titlefootnotes
\phfnote@mympfootnotemark
\phfnote@mpfootnoteglue

Some formatting utilities which can be overridden if you know what you're doing. \phfnote@fmt@titlefootnotes allows you to override the font in which the title-footnotes/thanks are typeset. \phfnote@mympfootnotemark is responsible for formatting its argument as a footnote mark, usually in superscript. \phfnote@mpfootnoteglue is the glue which is used between two footnote texts (as they are typeset in a single paragraph).

```
\def\phfnote@mpfootnotes@fontparsetup{%
  \parshape 1 0.04\textwidth 0.96\textwidth\relax}
\def\phfnote@fmt@titlefootnotes{}
\def\phfnote@mympfootnotemark#1{\@textsuperscript{\normalfont#1}}
\def\phfnote@mpfootnoteglue{\hskip 1.2em plus 2em minus 0.5em\relax}
```

For those not using the main 'default' title style

We use LaTeX's own \thanks mechanism, however we patch on the possibility for using \thanks[N]{text} and \thanksmark[N] for overriding the number which is used.

\notetitle@thanksmark

The \thanksmark is trivially implemented by \footnotemark. Very handy indeed.

Again, this macro is only made available as \thanksmark when a title style is set in \phfnote@do@notetitle.

```
\def\notetitle@thanksmark{\footnotemark}
```

Start by saving the old \thanks macro, just in case.

```
\let\phfnote@old@thanks\thanks
```

\notetitle@thanks

Now, we need to extend LaTeX's \thanks to allow an optional argument as for footnotes. This macro will be renamed \thanks in \phfnote@do@notetitle.

Check whether there is an optional argument; if there is none we execute LaTeX's original thanks code (replicated here), otherwise, we specify the optional argument explicitly at the relevant location in LaTeX's implementation:

```
\def\notetitle@thanks{\@ifnextchar[\phfnote@thanks{\phfnote@thanks[]}}%]
\long\def\phfnote@thanks[#1]#2{%
  \if\relax\detokenize{#1}\relax%
```

The optional argument is empty—just execute LaTeX's original \thanks code, replicated here:

```
    \footnotemark%
    \protected@xdef\@thanks{\@thanks\protect\footnotetext[\the\c@footnote]{#2}}%
```

Otherwise, execute LaTeX's original \thanks code, but with the optional argument inserted wherever needed:

```
  \else% argument, pass on to sub-commands:
    \footnotemark[#1]%
    \protected@xdef\@thanks{\@thanks\protect\footnotetext[#1]{#2}}%
  \fi%
}
```

5.2.3 Title Styles Definition

The title styles are documented in subsubsection 4.1.1.

Title style: 'default'

Implementation our main 'default' title style. See subsubsection 4.1.1.

\notetitleinnervsep Controls the vertical spacing between individual elements of the title.

```
\newcommand\notetitleinnervsep{1.15ex}
```

\notetitlewidth Controls the width of the area in which the title content is typeset. For more complex titles (e.g., pretty style, the title is typeset in a smaller width than the text width to allow room for decorations).

```
\def\notetitlewidth{\textwidth}
```

\notetitleparskip The paragraph skip that is used if the title contains multiple paragraphs.

```
\def\notetitleparskip{1.4ex}% parskip for multiple pars in main title
```

\notetitlefontparsetup
\notetitleaftertitleskip \notetitlefontparsetup sets up any necessary LaTeX commands to typeset the main title. This should set the font size, and then maybe \centering, a \parshape, or a text color.

```
\def\notetitlefontparsetup{\raggedright\setstretch{1.05}\Large}
```

\notetitleaftertitleskip generates the spacing after the main title. The default implementation behaves differently whether the title was multi-paragraph or not.

```
\def\notetitleaftertitleskip{%
  \ifnotetitle@default@ismultipar
    \vspace{\parskip}%
    %\gdef\phfnote@tmp@nextskip{\z@}%
    \gdef\phfnote@tmp@nextskip{0.5\dimexpr\notetitleinnervsep\relax}%
  \else
    \gdef\phfnote@tmp@nextskip{\notetitleinnervsep}%
  \fi
}
```

\notetitleauthorfontparsetup
\notetitledatefontparsetup

These macros set up any necessary LaTeX commands to typeset the author and the date. This should set the font size, and then maybe \centering, a \parshape, or a text color. These should use calls to \notetitledontextvskip to adjust vertical spacing between the title items.

```
\def\notetitleauthorfontparsetup{%
  \notetitledonextvskip[2]%
  \parshape 1 0.04\textwidth 0.96\textwidth\relax
  \strut
}
\def\notetitledatefontparsetup{%
  \notetitledonextvskip
  \parshape 1 0.04\textwidth 0.96\textwidth\relax
  \strut
}
```

\notetitledonextvskip

This helper macro is not meant to be redefined, but rather invoked from \notetitleauthorfontparsetup and \notetitledatefontparsetup. It adds the vertical space that was stored in \phfnote@tmp@nextskip.

```
\newcommand\notetitledonextvskip[1][]{%
  \vspace{#1\dimexpr\phfnote@tmp@nextskip\relax}%
  \gdef\phfnote@tmp@nextskip{\notetitleinnervsep}%
}
```

\notetitlemakecontents

Helper command that produces the content of the title. The default implementation uses the set \title, \author, and \date to render everything nicely. Note that if you redefine this, then it's up to you to honor what to do with \notetitlefontparsetup, \notetitlefont, \notetitleauthorfont, etc.

The default implementation of \notetitlemakecontents allows \notetitle*fontparsetup to take a single argument, the whole title/author/date including necessary formatting. In that case make sure to enclose that argument in a group.

```
\newcommand\notetitlemakecontents{
  \notetitlemakecontentstop
  {\par
    \let\phfnote@old@par\par
    \notetitle@titledefault@preparetitle
    \expandafter\notetitlefontparsetup\expandafter{%
      \expandafter\notetitlefont\expandafter{\@title}}%
    \phfnote@old@par
    \notetitleaftertitleskip
  }%
  \if\relax\detokenize\expandafter{\@author}\relax\else
    \expandafter\notetitleauthorfontparsetup\expandafter{%
      \expandafter\notetitleauthorfont\expandafter{\@author}}\par
  \fi
  \if\relax\detokenize\expandafter{\@date}\relax\else
    \expandafter\notetitledatefontparsetup\expandafter{%
      \expandafter\notetitledatefont\expandafter{\@date}}\par
  \fi
  \notetitlemakecontentsbottom
}
```

\notetitlemakecontentstop
\notetitlemakecontentsbottom

If you just want to insert stuff before/after in the title box, then you don't have to redefine all of \notetitlemakecontents, you can simply redefine these macros to whatever you like.

```
\def\notetitlemakecontentstop{}
\def\notetitlemakecontentsbottom{}
```

\notetitlebeginrender
\notetitleendrender

The title is rendered enclosed by calls to these macros. By default, render the title in a minipage.

WARNING: An important thing to note is that if you use \notetitle@default@usesavebox, then the saved box is an \hbox, not a \vbox (more flexibility). That means that the \notetitlebeginrender and \notetitleendrender commands must open and close some environment (or a \vbox or something like that) so that the whole \notetitlebeginrender...\notetitleendrender construction results in something that can be placed in an \hbox.

```
\def\notetitlebeginrender{\begin{minipage}{\notetitlewidth}}
\def\notetitleendrender{\end{minipage}}
```

\notetitleusemainbox

This macro is the main formatting command for the title's global appearance. After the title is typeset in a TeX box register, this macro is invoked to actually display it. The default title style simply displays it with a rule below, but the pretty title style does fancier things. Note this is only used if \ifnotetitle@default@usesavebox is true.

```
\newcommand\notetitleusemainbox[1]{%
  \par
  \box#1%
  \vspace*{\notetitleinnervsep}%
  \notetitlehrule\relax
}
```

\notetitle@default@mainbox

The main box register in which the title contents is saved.

```
\newsavebox\notetitle@default@mainbox
```

\ifnotetitle@default@usesavebox

This if controls whether or not we set up the title in a temporary TEX box register first, before displaying it. This allows to play around with the box, measure its height/width, place it into graphics, etc. But if we want a simple title, this is not necessary, and it might break some more fragile constructions (footnotes, etc).

If this is true, then the macro \notetitleusemainbox is called to render the box. Otherwise, \notetitleusemainbox is not called and the title is rendered directly.

```
\newif\ifnotetitle@default@usesavebox
\notetitle@default@usesaveboxtrue
```

\ifnotetitleusempfootnotes

Whether any \thanks commands in the title generate footnotes inside the minipage (i.e., a line at the bottom with e.g. affiliations), or whether they appear as regular footnotes at the bottom of the page.

This conditional may only be set to true if the rendering happens in a minipage. The minipage must be opened in \notetitlebeginrender and closed in \notetitleendrender.

```
\newif\ifnotetitleusempfootnotes
\notetitleusempfootnotestrue
```

\notetitle@default@setup

Any additional setup to be done at the beginning.

```
\def\notetitle@default@setup{%
    \notetitle@default@ismultiparfalse
    \gdef\phfnote@tmp@nextskip{\z@}%
    \par\raggedright}
```

Now we have the main implementation of the default title style.

\notetitle@style@default

A default title style, providing a flexible engine with powerful customization features (the same engine is used for the pretty title style).

The strategy goes like this: First typeset everything in a minipage enclosed in a box register, and then display that box register using \notetitleusemainbox.

```
\newcommand{\notetitle@style@default}{%
  \begingroup
    \parskip=\z@\relax
    \parindent=\z@\relax
    \providecommand\singlespace{}%
    \notetitle@default@setup
    \ifnotetitleusempfootnotes
      \phfnote@setupthanksmpfootnote
    \fi
    \vspace*{\notetitletopspace}%
    \def\x{}%
```

Now, draw the title (either in a box, or directly). If we save in a box, use an \hbox, not a \vbox, because we get size problems otherwise. See \notetitlebeginrender and \notetitleendrender.

```
    \ifnotetitle@default@usesavebox
      \def\x{\setbox\notetitle@default@mainbox=\hbox\bgroup}
    \fi
    \x\notetitlebeginrender
    \begingroup
          \singlespace%
          \notetitlemakecontents\par
          \ifnotetitleusempfootnotes
            \expandafter\ifstrequal\expandafter{\@mpfn}{mpfootnote}{}{%
              \PackageError{phfnote}{phfnote title: can only have
                'usempfootnotes' in a minipage}{Make sure you open a
                \string\begin{minipage} in the definition of
                  \string\notetitlebeginrender \space and correspondingly close
                  it with \string\end{minipage} in \string\notetitleendrender}%
            }
            \global\let\@thanks\@empty
            \phfnote@finalizempfootnotes{\phfnote@tmp@nextskip}%
          \fi
    \endgroup
    \notetitleendrender
    \def\x{}%
    \ifnotetitle@default@usesavebox
      \def\x{\egroup
        \notetitleusemainbox{\notetitle@default@mainbox}}%
    \fi
    \x
    \par
  \endgroup
  \vskip\notetitlebelowspace\relax% don't change this, abstract needs to \removelastskip
}
```

Some helpers for the default title style.

\ifnotetitle@default@ismultipar

This flag registers whether or not the title has multiple paragraphs (and thus renders on several spaced lines). It is set to true by redefining the \par command in the title (see \notetitle@titledefault@preparetitle below).

```
\newif\ifnotetitle@default@ismultipar
```

\notetitle@titledefault@preparetitle

This helper sets everything up to display the title. It redefines \par to register that the title has several paragraphs. Also, note that \parskip is inserted in \leavevmode, so that setting \parskip as below only affects subsequent paragraphs.

To change the parskip amount, you may simply patch this command, e.g., as `\appto\notetitle@titledefault@preparetitle{\parskip=1.2ex\relax}`.

```
\newcommand\notetitle@titledefault@preparetitle{%
  \def\par{\phfnote@old@par\global\notetitle@default@ismultipartrue}%
  \leavevmode\parskip=\notetitleparskip\relax}
```

Title style: ‘defaultv1’

`\notetitle@style@defaultv1` The default title style, copied from phfnote v1.0. DO NOT CHANGE. Kept only for backwards compatibility, in case someone had spent lots of time fine-tuning their title and patching the v1 of this package, or if they want the document to appear exactly as the v1 of the package.

[We need to use `\csdef` because of the “1” in the title name, which is not valid in an TeX usual escape sequence (!!)]

```
\csdef{notetitle@style@defaultv1}{%
  \begingroup\par\raggedright%
    \phfnote@setupthanksmpfootnote%
    \vspace*{\notetitletopspace}%
    \phfnote@title@checksetspace{defaultv1}%
    \begin{minipage}{\textwidth}%
      \begin{singlespace}%
        \parskip=0pt\parindent=0pt\relax%
        {\let\phfnote@old@par\par%
          \def\par{\phfnote@old@par%
            \parskip=1.5ex\relax\parshape 1 0pt \textwidth\relax%
            \noindent}%
          \par%
          \Large  {\notetitlefont \@title}\par}%
        \vskip 2mm\relax
        \if\relax\detokenize\expandafter{\@author}\relax\else%
          \par\parshape 1 0.04\textwidth 0.96\textwidth\relax%
          {\notetitleauthorfont \@author}%
          \vskip 2mm\relax%
        \fi
        \if\relax\detokenize\expandafter{\@date}\relax\else%
          \par\parshape 1 0.04\textwidth 0.96\textwidth\relax%
          {\notetitledatefont \@date}
          \vskip 2mm\relax%
        \fi
        \global\let\@thanks\@empty%
        \csname phfnote@finalizempfootnotes@v1\endcsname%
      \end{singlespace}%
    \end{minipage}\par%
    \vspace*{2mm}%
    \notetitlehrule\relax%
    \par%
  \endgroup%
  \vskip\notetitlebelowspace\relax% don’t change this, abstract needs to \removelastskip
}
\csdef{phfnote@finalizempfootnotes@v1}{%
  \global\setbox\@mpfootins=\vbox{%
    \parskip=0pt\parindent=0pt\parshape 1 0.04\textwidth 0.96\textwidth\relax%
    \noindent\leavevmode%
    \reset@font\footnotesize%
    \phfnote@fmt@titlefootnotes%
    \phfnote@mpfootmaterial}%
}
\def\phfnote@title@checksetspace#1{%
  \ifdefined\singlespace\else%
    \PackageError{phfnote}{Note title style ‘#1’ requires the
      ‘setspace’ package to be loaded!  Please load it, or use a
      pkgset which loads it automatically}{}%
  \fi%
}
```

Title style: ‘pretty’ and ‘pretty2’

The pretty & pretty2 styles uses the same engine as the default, with different settings.

\notetitle@style@pretty
\notetitle@style@pretty2

Alias the main maketitle engine to the default one.

```
\let\notetitle@style@pretty\notetitle@style@default
\cslet{notetitle@style@pretty2}\notetitle@style@default
```

\notetitle@stylesetup@pretty

Setup macro that makes everything nice. Do all these defs in the macro definition, and not directly in the global space, to avoid polluting the LaTeX environment with useless definitions if we don’t use this title style.

```
\def\notetitle@stylesetup@pretty{%
  \RequirePackage{xcolor}
  \long\def\notetitlefont##1{\bfseries ##1}
  \def\notetitlefontparsetup{%
    \color{notetitleprettytextcolor}\centering}
  \def\notetitleauthorfontparsetup{%
    \notetitledonextvskip[2]%
    \color{notetitleprettytextcolor}\centering}
  \def\notetitledatefontparsetup{%
    \notetitledonextvskip
    \color{notetitleprettytextcolor}\centering}
  \def\phfnote@mpfootnotes@fontparsetup{\color{notetitleprettytextcolor}}
  %
  \def\notetitlewidth{\dimexpr\textwidth
    -\notetitleprettylsiderulewidth
    -\notetitleprettyrsiderulewidth
    -\notetitleprettylsidespacewidth
    -\notetitleprettyrsidespacewidth\relax}
  \let\notetitleusemainbox\notetitle@pretty@usemainbox
  %
  \def\notetitleprettylsiderulewidth{10pt}
  \def\notetitleprettylsidespacewidth{10pt}
  \def\notetitleprettyrsiderulewidth{10pt}
  \def\notetitleprettyrsidespacewidth{10pt}
  \def\notetitleprettytopspace{10pt}
  \def\notetitleprettybottomspace{10pt}
  \def\notetitleprettytophrulewidth{0pt}
  \def\notetitleprettybottomhrulewidth{0pt}
  %
  \definecolor{notetitleprettylsiderulecolor}{RGB}{0,68,126}
  \colorlet{notetitleprettyrsiderulecolor}{notetitleprettylsiderulecolor}
  \colorlet{notetitleprettytophrulecolor}{notetitleprettylsiderulecolor}
  \colorlet{notetitleprettybottomhrulecolor}{notetitleprettylsiderulecolor}
  \definecolor{notetitleprettytextcolor}{RGB}{25,25,38}
  \colorlet{notetitleprettybgcolor}{white}
}
\newlength\notetitle@pretty@tmplenht
\newlength\notetitle@pretty@tmplendp
\def\notetitle@pretty@usemainbox#1{%
  \parskip=\z@\relax
  \parindent=\z@\relax
  \notetitle@pretty@tmplenht=\ht#1\relax%
  \notetitle@pretty@tmplendp=\dp#1\relax%
  \edef\tmp@dorule##1##2{%
    {\noexpand\color{notetitlepretty##1siderulecolor}%
      \noexpand\rule{##2}{%
        \dimexpr \notetitleprettytopspace+
            \notetitleprettybottomspace+
            \notetitleprettytophrulewidth+
            \notetitleprettybottomhrulewidth+
            \notetitle@pretty@tmplendp+
            \notetitle@pretty@tmplenht\relax}}}%
  \fboxsep=0pt% for \colorbox
  \par\hbox to \textwidth{%
    \hskip 0pt plus 0.1fil minus 0.1fil\relax%
    \tmp@dorule{l}{\notetitleprettylsiderulewidth}%
    \colorbox{notetitleprettybgcolor}{%
      \vbox{%
        {\color{notetitleprettytophrulecolor}%
          \hrule height \notetitleprettytophrulewidth\relax}%
        \hbox{%
          \hskip \notetitleprettylsidespacewidth\relax
          % \fbox% DEBUG
          {\vbox{\vskip \notetitleprettytopspace\relax
              \box#1%
              \vskip \notetitleprettybottomspace\relax}}%
          \hskip \notetitleprettyrsidespacewidth\relax
        }%
        {\color{notetitleprettybottomhrulecolor}%
          \hrule height \notetitleprettybottomhrulewidth\relax}%
      }}%
    \tmp@dorule{r}{\notetitleprettyrsiderulewidth}%
  \hskip 0pt plus 0.1fil minus 0.1fil\relax}%
  \par%
}
```

\notetitle@stylesetup@pretty2 The pretty2 style simply loads the pretty style, and ajusts some settings.

```
\csdef{notetitle@stylesetup@pretty2}{%
  \notetitle@stylesetup@pretty
  %
  \definecolor{notetitleprettylsiderulecolor}{RGB}{0,68,126}
  \colorlet{notetitleprettyrsiderulecolor}{notetitleprettylsiderulecolor}
  \colorlet{notetitleprettytophrulecolor}{notetitleprettylsiderulecolor}
  \colorlet{notetitleprettybottomhrulecolor}{notetitleprettylsiderulecolor}
  \colorlet{notetitleprettytextcolor}{notetitleprettylsiderulecolor!50!black}
  \colorlet{notetitleprettybgcolor}{white!95!notetitleprettytextcolor}
  %
  \def\notetitleprettytophrulewidth{.4pt}
  \def\notetitleprettybottomhrulewidth{.4pt}
}
```

Title style: ‘small’

Implementation an alternate ‘small’ title style.

\notetitle@style@small The small title style. Again, use the main title engine, but customize the settings.

```
\let\notetitle@style@small\notetitle@style@default
\newcommand\notetitle@stylesetup@small{%
  \notetitleusempfootnotesfalse
  %
  \def\notetitlemakecontents{%
    {\expandafter\notetitlefont\expandafter{\@title}}%
    \hfill\makebox{\fontsize{9pt}{10pt}\selectfont
      \notetitle@small@renderauthordate}%
  }
  %\notetitle@default@usesaveboxfalse
  %\def\notetitlebeginrender{\par}
  %\def\notetitleendrender{%
  %  \vspace*{\notetitleinnervsep}\notetitlehrule\relax\vspace*{\notetitleinnervsep}}
  \def\notetitleusemainbox##1{%
    \par\box##1%
    \vspace*{\notetitleinnervsep}\notetitlehrule\relax\vspace*{\notetitleinnervsep}}
  \def\notetitle@small@renderauthordate{%
    \expandafter\notblank\expandafter{\@author}{%
      \expandafter\notblank\expandafter{\@date}{% both not blank
        {\expandafter\notetitleauthorfont\expandafter{\@author}}%
        \notetitlesmallauthordatesep
        {\emph{\expandafter\notetitledatefont\expandafter{\@date}}}%
      }{% only author
        {\expandafter\notetitleauthorfont\expandafter{\@author}}%
      }%
    }{% only date
      {\emph{\expandafter\notetitledatefont\expandafter{\@date}}}%
    }}
  %
  \def\notetitleinnervsep{1mm}
  \def\notetitlesmallauthordatesep{\hspace*{2mm}--\hspace*{2mm}}
}
```

Style `smallv1` provided for backwards compatibility, to make sure that all spacing and formatting is exactly the same as for phfnote version 1.0.

```
\csdef{notetitle@style@smallv1}{%
  \begingroup\par\raggedright%
    \let\footnote\thanks%
    \vspace*{\notetitletopspace}%
    {\expandafter\notetitlefont\expandafter{\@title}}%
    \hfill\makebox{\fontsize{9pt}{10pt}\selectfont
      {\expandafter\notetitleauthorfont\expandafter{\@author}}%
      \hspace*{2mm}--\hspace*{2mm}{\emph{\expandafter\notetitledatefont\expandafter{\@date}}}
    \vspace*{1mm}\notetitlehrule\relax\vspace*{1mm}%
    \par%
  \endgroup%
  \vskip\notetitlebelowspace\relax% don't change this, abstract needs to \removelastskip
}
```

Title style: 'article'

Implementation the 'article' title style.

`\notetitle@style@article`
`\notetitle@stylesetup@article`

Technical note. Here, by using a `tabular` environment for authors, we need to assume that the macro `\notetitleauthorfont` takes one argument and does not leave the author contents surrounded with full braces. This would cause '\\' tokens enclosed in the protective braces to cause errors in the tabular environment.

```
\let\notetitle@style@article\notetitle@style@default
\newcommand{\notetitle@stylesetup@article}{
  %\def\notetitletopspace{-3em}
  \def\notetitlebottomspace{2.5em}
  \def\notetitleinnervsep{1.5em}
  \def\notetitlefont{}
  \def\notetitlefontparsetup{%
    \LARGE\centering}
  \long\def\notetitleauthorfontparsetup##1{%
```

```
    \notetitledonextvskip%
    {\large\centering
    \lineskip .5em\relax%
    \begin{tabular}[t]{c}%
      ##1%
    \end{tabular}\par}}
  \long\def\notetitleauthorfont##1{\large ##1}
  \def\notetitledatefontparsetup{%
    \notetitledonextvskip
    \centering}
  \def\phfnote@mpfootnotes@fontparsetup{}
  \def\notetitleusemainbox##1{%
    \par
    \box##1%
  }
  \appto\notetitle@default@setup{%
    %\def\singlespace{}%
  }
  %
  \notetitleusempfootnotesfalse
}
```

\notetitle@style@articlev1 Backwards compatibility style articlev1.

```
\csdef{notetitle@style@articlev1}{%
  \vspace*{-3em}%
  \begingroup
    \centering
    \let\footnote\thanks%
    {\LARGE \@title \par}%
    \vskip 1.5em%
    {\large%
      \lineskip .5em%
      \begin{tabular}[t]{c}%
        \@author%
      \end{tabular}\par}%
    \vskip 1.5em%
    {\large \@date}%
    \par%
  \endgroup%
  \par%
  \vskip 2.5em\relax%
}
```

5.2.4 Plugging into \maketitle

Actually perform the definitions to make \maketitle produce the title with the given style. Specifically, we override \@maketitle. The latter is called internally by \maketitle, and the advantage of overriding \@maketitle only is that we inherit the mechanism provided by the original class (e.g., `article`) to deal with two-column layouts.

`\phfnote@do@notetitle` This macro takes care of installing the correct title into the document, by overriding `\@maketitle`.

This macro is called later after processing the package options. Its argument `#1` is the style name, e.g., `default`.

```
\def\phfnote@do@notetitle#1{
```

If we have the title style `false`, or an empty title style, then we leave default title provided by the class.

```
  \ifstrequal{#1}{false}{}{%
    \if\relax\detokenize\expandafter{#1}\relax
    \else
```

Otherwise, we have a title style to set. Do some checks that the given style is indeed defined.

```
      \ifcsname notetitle@style@#1\endcsname
        \def\phfnote@tmp@titsty{#1}%
      \else
        \PackageError{phfnote}{Unknown title style: '#1'.}{Unknown title
          style: '#1'. Please consult the package documentation for available
          styles.}
        \def\phfnote@tmp@titsty{default}%
      \fi
```

Apply new (default) definitions of `\thanks`, `\thanksmark` and `\title`. Do this here only, because this can clash with more complicated versions from, e.g., RevTeX.

```
      \let\title\notetitle@title
      \let\thanks\notetitle@thanks
      \let\thanksmark\notetitle@thanksmark
```

Also, LaTeX initializes `\@author` with code that generates a warning that no author is given. We don't want that. It's perfectly fine not to have an author, and in this case this must be empty so that our title routines can properly handle this case.

```
      \def\@author{}%
```

Now, actually overload the title style by redefining `\@maketitle`. Also, if the style defines a "setup" macro `\notetitle@stylesetup@...`, then we invoke it.

```
      \ifcsname notetitle@stylesetup@\phfnote@tmp@titsty\endcsname
        \csname notetitle@stylesetup@\phfnote@tmp@titsty\endcsname
      \fi
      \def\@maketitle{\csname notetitle@style@\phfnote@tmp@titsty\endcsname}%
    \fi
  }%
}
```

5.3 Abstract

Now we can take care of the abstract. Unlike the title styles, the abstract has a base implementation. Then, we may have attributes which change some parameters.

`notedefaultabstract` First, save the old environment `\begin{abstract}...\end{abstract}` provided by the class (if any).

```
\let\notedefaultabstract\abstract
\let\endnotedefaultabstract\endabstract
```

`\noteabstracttextfont` `\noteabstractnamefont` `\noteabstracttextwidth` `\noteabstractafterspacing` `\noteabstractbeforepacing` Macros which can be overridden to customize the abstract. See subsection 4.2.

```
\newcommand{\noteabstracttextfont}{}
\newcommand{\noteabstractnamefont}{\bfseries\small}
\if@twocolumn
  \newcommand\noteabstracttextwidth{\hsize}
\else
  \newcommand{\noteabstracttextwidth}{0.9\hsize}
\fi
\newcommand\noteabstractafterspacing{1.5em}
\newcommand\noteabstractbeforespacing{1.5em}
```

`\noteabstract@nameline` Create the line which contains the title of the abstract, that is, the word “Abstract.” This can be overloaded, of course, for customization.

```
\def\noteabstract@nameline{
  {\parskip=0pt\relax\par\centering\noteabstractnamefont%
    \abstractname%
    \par}\vskip 1ex\relax%
}
```

`noteabstract` The proper `noteabstract` environment.

```
\newenvironment{noteabstract}{%
  \removelastskip%
  \vspace{\noteabstractbeforespacing}%
  \begingroup%
    \par\noindent\centering%
    \begin{minipage}{\noteabstracttextwidth}%
      \noteabstract@nameline%
      \noteabstracttextfont%
    }%
    {%
    \end{minipage}%
    \par%
  \endgroup%
  \vspace{\noteabstractafterspacing}%
}
```

The abstract can be customized by the attributes. Here we define them:

```
\def\noteabstract@attr@wide{%
  \def\noteabstracttextwidth{\textwidth}%
}
\def\noteabstract@attr@narrow{%
  \if@twocolumn
  \else
    \def\noteabstracttextwidth{0.8\textwidth}%
  \fi
}
\def\noteabstract@attr@noname{%
  \def\noteabstract@nameline{}%\vspace*{1ex}}%
}
\def\noteabstract@attr@original{%
  \let\abstract\notedefaultabstract
  \let\endabstract\endnotedefaultabstract
}
\def\noteabstract@attr@small{%
  \g@addto@macro\noteabstracttextfont{\small}%
}
\def\noteabstract@attr@compact{%
  \renewcommand\noteabstractafterspacing{1ex}%
  \renewcommand\noteabstractbeforespacing{1ex}%
}
\def\noteabstract@attr@it{%
  \g@addto@macro\noteabstracttextfont{\itshape}%
}
```

\phfnote@do@noteabstract This helper both defines the abstract environment, and also sets the abstract attributes. This macro will be called according to the package options.

#1 = a comma-separated list of attributes, or the string false.

```
\def\phfnote@do@noteabstract#1{%
  \ifstrequal{#1}{false}{}{%
    \let\abstract\noteabstract
    \let\endabstract\endnoteabstract
    \phfnote@internal@execattribs{noteabstract@attr@}{abstract attribute}{#1}
  }%
}
```

5.4 Page Geometry Settings

For the page geometry settings, we just have a bunch of styles which we define as macros. The macros just set up \PassOptionsToPackage for the geometry package. Then the correct macro will be selected according to the current phfnote package options.

The description of these settings are given in subsection 4.5.

\phfnote@pagegeomstyle@default Default setting.

```
\def\phfnote@pagegeomstyle@default{
  \if@twocolumn
    \PassOptionsToPackage{hmargin=1in,vmargin=0.75in,includeheadfoot}{geometry}%
  \else
    % fix the margins a bit to make text wider
    \ifcase\@ptsize% mods for 10 pt
      \PassOptionsToPackage{hmargin=1.5in,vmargin=1.25in}{geometry}%
    \or% mods for 11 pt
      \PassOptionsToPackage{hmargin=1.5in,vmargin=1.25in}{geometry}%
    \or% mods for 12 pt
      \PassOptionsToPackage{hmargin=1.25in,vmargin=1.25in}{geometry}%
    \fi%
  \fi
}
```

\phfnote@pagegeomstyle@narrow Narrow style.

```
\def\phfnote@pagegeomstyle@narrow{
  \if@twocolumn
    \PassOptionsToPackage{hmargin=1.25in,vmargin=0.75in,includeheadfoot}{geometry}%
  \else
    % fix the margins a bit to make text wider
    \ifcase\@ptsize% mods for 10 pt
      \PassOptionsToPackage{hmargin=1.75in,vmargin=1.5in}{geometry}%
    \or% mods for 11 pt
      \PassOptionsToPackage{hmargin=1.75in,vmargin=1.5in}{geometry}%
    \or% mods for 12 pt
      \PassOptionsToPackage{hmargin=1.5in,vmargin=1.5in}{geometry}%
    \fi%
  \fi
}
```

\phfnote@pagegeomstyle@wide Wide style.

```
\def\phfnote@pagegeomstyle@wide{
  \if@twocolumn
    \PassOptionsToPackage{hmargin=0.75in,vmargin=0.75in,includeheadfoot}{geometry}%
  \else
    % fix the margins a bit to make text wider
    \ifcase\@ptsize% mods for 10 pt
      \PassOptionsToPackage{hmargin=1.25in,vmargin=1.25in}{geometry}%
    \or% mods for 11 pt
      \PassOptionsToPackage{hmargin=1.25in,vmargin=1.25in}{geometry}%
    \or% mods for 12 pt
      \PassOptionsToPackage{hmargin=1in,vmargin=1.25in}{geometry}%
    \fi%
  \fi
}
```

`\phfnote@pagegeomstyle@xwide`

Extra wide.

Changed in v3.0 [2018/11/30]: Changed the `xwide` page geometry for tighter vertical margin for single-column text (use `pagegeom=xwidev1` instead for old behavior).

```
\def\phfnote@pagegeomstyle@xwide{
  \if@twocolumn
    \PassOptionsToPackage{hmargin=0.5in,vmargin=0.5in,includeheadfoot}{geometry}%
  \else
    % fix the margins a bit to make text wider
    \ifcase\@ptsize% mods for 10 pt
      \PassOptionsToPackage{hmargin=1in,vmargin=1in}{geometry}%
    \or% mods for 11 pt
      \PassOptionsToPackage{hmargin=1in,vmargin=1in}{geometry}%
    \or% mods for 12 pt
      \PassOptionsToPackage{hmargin=0.75in,vmargin=1in}{geometry}%
    \fi%
  \fi
}
\csdef{phfnote@pagegeomstyle@xwidev1}{
  \if@twocolumn
    \PassOptionsToPackage{hmargin=0.5in,vmargin=0.5in,includeheadfoot}{geometry}%
  \else
    % fix the margins a bit to make text wider
    \ifcase\@ptsize% mods for 10 pt
      \PassOptionsToPackage{hmargin=1in,vmargin=1.25in}{geometry}%
    \or% mods for 11 pt
      \PassOptionsToPackage{hmargin=1in,vmargin=1.25in}{geometry}%
    \or% mods for 12 pt
      \PassOptionsToPackage{hmargin=0.75in,vmargin=1.25in}{geometry}%
    \fi%
  \fi
}
```

`\phfnote@pagegeomstyle@bigmargin`

`bigmargin` style.

```
\def\phfnote@pagegeomstyle@bigmargin{%
  \if@twocolumn
    \PassOptionsToPackage{hmargin=1.5in,vmargin=0.75in,includeheadfoot}{geometry}%
  \else
    % fix the margins a bit to make text wider
    \ifcase\@ptsize% mods for 10 pt
      \PassOptionsToPackage{hmargin={2.25in,1.75in},vmargin=1.25in}{geometry}%
    \or% mods for 11 pt
      \PassOptionsToPackage{hmargin={2.25in,1.75in},vmargin=1.25in}{geometry}%
    \or% mods for 12 pt
      \PassOptionsToPackage{hmargin={2in,1.5in},vmargin=1.25in}{geometry}%
    \fi%
  \fi
}
```

\phfnote@do@pagegeom

Finally, provide a helper to set the page geometry. Just call the right macro. If the argument is `false`, don't do anything.

```
\newcommand{\phfnote@do@pagegeom}[1]{
  \ifstrequal{#1}{false}{}{%
    \message{phfnote: Setting page geometry style #1}%
    \ifcsname phfnote@pagegeomstyle@#1\endcsname
      \csname phfnote@pagegeomstyle@#1\endcsname
    \else
      \PackageWarning{phfnote}{Unknown page geometry style: ‘#1’!}%
    \fi
    %
    \RequirePackage{geometry}%
  }%
}
```

5.5 Text, Paragraph and Line Spacing

Text & Line Spacing

\phfnote@do@spacing

Some cosmetic definitions to adjust line spacing. The line spacing is slightly adjusted according to font size to make the document more readable. Depending on whether the setspace package is loaded, we use it or go low-level with a redefinition of LaTeX' `\baselinestretch`. If the captions package is loaded, the figure captions' line spacing is also adjusted.

Also set an `\emergencystretch` so that lines get spaced out for underfull boxes, rather than overflowing far into the margin.

```
\def\phfnote@do@spacingdefs#1{
  \ifstrequal{#1}{false}{}{%
    \@ifpackageloaded{setspace}{
      \def\phfnote@dostretch##1{%
        \setstretch{##1}\phfnote@docaptionstretch{##1}}
    }{
      \def\phfnote@dostretch##1{%
        \renewcommand\baselinestretch{##1}\phfnote@docaptionstretch{##1}}
    }
    \@ifpackageloaded{caption}{
      \def\phfnote@docaptionstretch##1{\captionsetup{font={stretch=##1}}}
    }{
      \def\phfnote@docaptionstretch##1{\PackageWarning{phfnote}{Can't
          set line spacing for captions, because the package `caption'
          is not loaded.  Please load it before `phfnote', or use an
          appropriate pkgset (e.g. `rich') which loads this package
          automatically.}}
    }
    \if@twocolumn
      \phfnote@dostretch{1.0} % leave default
      \emergencystretch=3em\relax
    \else
      \ifcase\@ptsize% 10pt
        \phfnote@dostretch{1.1}
      \or% 11pt
        \phfnote@dostretch{1.0} % 1.05? better 1.0...
      \or% 12pt
        \phfnote@dostretch{1.0} % 1.03? not really noticeable...
      \fi
      \emergencystretch=6em\relax
    \fi
  }
}
```

Paragraph Spacing Presets

Here again, we define several possibilities for paragraph settings as individual macros (see subsection 4.7). Depending on the package option, we execute the corresponding macro.

```
\def\phfnote@par@original{%
}
\def\phfnote@par@indent{%
  \parindent=1.5em\relax
  \parskip=0pt\relax
}
\def\phfnote@par@indentminiskip{%
  \parindent=1.5em\relax
  \parskip=0.3em plus 0.1em\relax
}
\def\phfnote@par@skip{%
  \parindent=0pt\relax
  \parskip=0.8em plus 0.2em minus 0.1em\relax
}
```

\phfnote@do@par Execute the given paragraph setting. The argument #1 is the setting, for example, skip.

```
\def\phfnote@do@par#1{%
  \ifstrequal{#1}{false}{}{%
    \ifcsname phfnote@par@#1\endcsname
      \csname phfnote@par@#1\endcsname
    \else
      \PackageWarning{phfnote}{Bad paragraph setting: #1. Leaving original}
    \fi
  }
}
```

5.6 Section Styling

Very limited support for styling section and paragraph headers (subsection 4.6). If you want anything serious, use sectsty or titlesec directly.

\notesectionallfont
\notesectionallfontfamily

Define the \notesectionallfont and \notesectionallfontfamily, which control the general font used in section headings.

```
\newcommand{\notesectionallfont}{%
  \fontfamily{\notesectionallfontfamily}\fontseries{bx}\selectfont}
\newcommand{\notesectionallfontfamily}{ppl}
```

\notesectionfont
\notesubsectionfont
\notesubsubsectionfont
\noteparagraphfont
\notesubparagraphfont

These macros are called for their respective sectioning command, after \notesectionallfont has been invoked. (Again, only for those sectioning commands which are styled by us.)

```
\newcommand{\notesectionfont}{\large}
\newcommand{\notesubsectionfont}{\normalsize}
\newcommand{\notesubsubsectionfont}{\small}
\newcommand{\noteparagraphfont}{\normalsize}
\newcommand{\notesubparagraphfont}{\normalsize}
```

\notesectionsetfonts
\noteparagraphsetfonts

Helpers to directly set the font commands for \section, \subsection and \subsubsection (with \notesectionsetfonts), and for \paragraph and \subparagraph (with \noteparagraphsetfonts).

```
\newcommand{\notesectionsetfonts}[3]{%
  \renewcommand{\notesectionfont}{#1}%
  \renewcommand{\notesubsectionfont}{#2}%
  \renewcommand{\notesubsubsectionfont}{#3}%
}
\newcommand{\noteparagraphsetfonts}[2]{%
  \renewcommand{\noteparagraphfont}{#1}%
  \renewcommand{\notesubparagraphfont}{#2}%
}
```

Define the attributes which the user can set. See subsection 4.6.

```
\def\phfnote@do@secfmt@section{
  \RequirePackage{sectsty}
  \sectionfont{\notesectionallfont\notesectionfont}
  \subsectionfont{\notesectionallfont\notesubsectionfont}
  \subsubsectionfont{\notesectionallfont\notesubsubsectionfont}
}
\def\phfnote@do@secfmt@paragraph{
  \RequirePackage{sectsty}
  \paragraphfont{\notesectionallfont\noteparagraphfont}
  \subparagraphfont{\notesectionallfont\notesubparagraphfont}
}
\def\phfnote@do@secfmt@compact{
  \notesectionsetfonts{\normalsize}{\small}{\small}
}
\def\phfnote@do@secfmt@larger{
  \notesectionsetfonts{\Large}{\large}{\normalsize}
}

\def\phfnote@do@secfmt@secsquares{
  \RequirePackage{amssymb}
  \let\phfnote@secsquares@old@seccntformat\@seccntformat
  \def\@seccntformat##1{%
    \expandafter\ifx\csname ##1\endcsname\section\relax%
    \unexpanded{\makebox[0pt][r]{\raisebox{0.15ex}{{%
            \notesmaller[0.6]\ensuremath{\blacksquare}}}%
        \hspace*{1.2ex}}}%
    \fi%
    \phfnote@secsquares@old@seccntformat{##1}}
}
\def\phfnote@do@secfmt@secnummargin{
  \let\phfnote@secnummargin@old@seccntformat\@seccntformat
  \def\@seccntformat##1{%
    \protect\makebox[0pt][r]{\phfnote@secnummargin@old@seccntformat{##1}}}
}

\def\phfnote@do@secfmt@rmfamily{
  \renewcommand\notesectionallfontfamily{\rmdefault}
}
\def\phfnote@do@secfmt@sffamily{
  \renewcommand\notesectionallfontfamily{\sfdefault}
}
\def\phfnote@do@secfmt@itpar{
  \def\noteparagraphfont{\normalfont\normalsize\itshape}
  \def\notesubparagraphfont{\normalfont\normalsize\itshape}
}
\def\phfnote@do@secfmt@blockpar{
  \let\phfnote@old@paragraph\paragraph
  \def\paragraph##1{%
    \phfnote@old@paragraph{##1}%
    \hspace*{0pt}\par\nopagebreak% ugly hack!!
  }
}
```

\phfnote@do@secfmt

Actually perform the required styling, according to the package options given as argument. The argument is a comma-separated list of attributes specified by the user, or the string `false`.

```
\def\phfnote@do@secfmt#1{%
  \ifstrequal{#1}{false}{}{%
    \phfnote@internal@execattribs{phfnote@do@secfmt@}{section formatting preset}{#1}%
  }
}
```

5.7 LaTeX Package Sets

Define the package sets as macros. Depending on the user-specified options we load the corresponding one(s) (several may be specified).

See subsection 4.4 for a description of what these package sets do.

\phfnote@do@pkgset@none
\phfnote@do@pkgset@minimal
\phfnote@do@pkgset@rich
\phfnote@do@pkgset@extended

Macros which implement the package sets. Each macro invokes `\RequirePackage` for the appropriate packages.

```
\def\phfnote@do@pkgset@none{
}

\def\phfnote@do@pkgset@minimal{

  \RequirePackage{amsmath}
  \RequirePackage{amsfonts}
  \RequirePackage{amssymb}
  \RequirePackage{amsthm}

  \RequirePackage{xcolor}

}


\def\phfnote@internal@setifxeorlua#1#2{%
  \ifXeTeX\let#1#2\fi
  \ifLuaTeX\let#1#2\fi
}

\def\phfnote@do@pkgset@rich{

  \phfnote@do@pkgset@minimal

  \RequirePackage{setspace}
  \RequirePackage{caption}

  \PassOptionsToPackage{shortlabels}{enumitem}
  \RequirePackage{enumitem}

  \RequirePackage{graphicx}

```

For this bit, use the iftex package to determine if we're running XeTeX or LuaTeX; if that's the case then we inhibit the loading of inputenc and fontenc. The `\IfFileExists` is to ensure the package runs on older LaTeX distributions without iftex.

Plus, load inputenc, resp. fontenc, only if it isn't already loaded.

Changed in v1.1 [2018/08/27]: If running XeTeX or LuaTeX, then do not load inputenc and fontenc as part of `rich` and `extended` package sets. Plus, do not load inputenc (resp. fontenc) if the package is already loaded.

```
  \def\phfnote@tmp@requireinputencfontenc{
    \@ifpackageloaded{fontenc}{}{
      \PassOptionsToPackage{T1}{fontenc}
      \RequirePackage{fontenc}
    }
    \@ifpackageloaded{inputenc}{}{
      \PassOptionsToPackage{utf8}{inputenc}
      \RequirePackage{inputenc}
    }
  }
  \IfFileExists{iftex.sty}{
    \RequirePackage{iftex}
    \phfnote@internal@setifxeorlua\phfnote@tmp@requireinputencfontenc\relax
  }{}
  \phfnote@tmp@requireinputencfontenc

```

Load microtype after fontenc (just in case).

```
  \RequirePackage{microtype}
}

\def\phfnote@do@pkgset@extended{

  \phfnote@do@pkgset@rich

  \RequirePackage{float}

  \RequirePackage{verbdef}

  \PassOptionsToPackage{autostyle,autopunct=true}{csquotes}
  \RequirePackage{csquotes}

  \RequirePackage{dsfont}
  \RequirePackage{bbm}
  \RequirePackage{mathtools}

}
```

\phfnote@do@pkgset Finally, define the helper which will load the required package sets.

```
\def\phfnote@do@pkgset#1{
  \phfnote@internal@execattribs{phfnote@do@pkgset@}{package set}{#1}
}
```

5.8 Hyperref Support and Hyperlinks

NOTE

The name ‘docnotelinkcolor’ is historical and hard-coded in many other files I’ve used, so I’m DEFINITELY NOT changing it.

Helpers—default set of hyperref options, and other helper macros.

```
\def\phfnote@hyperrefdefs@val@options{%
  bookmarksnumbered=false,bookmarksopen=false,bookmarksopenlevel=1,%
  breaklinks=true,pdfborder={0 0 0},colorlinks=true,%
  anchorcolor=docnotelinkcolor,citecolor=docnotelinkcolor,%
  filecolor=docnotelinkcolor,linkcolor=docnotelinkcolor,%
  menucolor=docnotelinkcolor,runcolor=docnotelinkcolor,%
  urlcolor=docnotelinkcolor%
}%
\def\phfnote@hyperrefdefs@deferredhypersetup#1{%
  \AtBeginDocument{%
    \@ifpackageloaded{hyperref}{%
      \hypersetup{#1}%
    }{%
      \PackageWarning{phfnote}{\MessageBreak\MessageBreak
        *** package ‘hyperref‘ was not loaded ***\MessageBreak
        Since you specified ‘hyperrefdefs=defer‘, I was expecting you would call
        ‘\string\usepackage{hyperref}‘ at some point later in your preamble, but
        it does not appear you did so.  Your document might look weird.}%
    }%
  }%
}
\providecommand\phfnote@hyperrefdefs@dopkgoptions{%
  \PassOptionsToPackage{unicode=true}{hyperref}
}
\def\phfnote@hyperrefdefs@loadhyperref{%
    \phfnote@hyperrefdefs@dopkgoptions
    \RequirePackage{hyperref}}
```

```
\def\phfnote@hyperrefdefs@provideemail{\let\email\phfnote@email}
\def\phfnote@eqref#1{%
  \hyperref[{#1}]{\textup{\tagform@{\ref*{#1}}}}%
}
\def\phfnote@hyperrefdefs@redefeqref{%
  \let\eqref\phfnote@eqref
}
\def\phfnote@hyperrefdefs@afterhook{}
```

Define the attributes that can be set for hyperref-related options. See subsection 4.11.

```
\def\phfnote@do@hyperrefdefs@attr@true{}% for explicit value "hyperrefdefs=true"
\def\phfnote@do@hyperrefdefs@attr@defer{
  \def\phfnote@hyperrefdefs@loadhyperref{%
    \phfnote@hyperrefdefs@dopkgoptions
    \let\hypersetup\phfnote@hyperrefdefs@deferredhypersetup}
}
\def\phfnote@do@hyperrefdefs@attr@clearoptions{
  \def\phfnote@hyperrefdefs@val@options{}
}
\def\phfnote@do@hyperrefdefs@attr@noemail{
  \def\phfnote@hyperrefdefs@provideemail{}
}
\def\phfnote@do@hyperrefdefs@attr@noeqref{
  \def\phfnote@hyperrefdefs@redefeqref{}
}
\def\phfnote@do@hyperrefdefs@attr@noload{
  \def\phfnote@hyperrefdefs@loadhyperref{}
  \def\phfnote@hyperrefdefs@redefeqref{}
  \let\hypersetup\@gobble
}
```

`\phfnote@do@hyperrefdefs`
`\email`
`\url`

Do all stuff related to hyperref.

```
\def\phfnote@do@hyperrefdefs#1{%
  \ifstrequal{#1}{false}{}{%
```

Make sure a color-managing package is loaded, color or xcolor, and define our default link color:

```
    \phfnote@requirecolorpackage%
    \definecolor{docnotelinkcolor}{rgb}{0,0,0.4}%
```

Load the url package, and save a version of `\url` which is not patched by hyperref to format URLs, and provide `\phfnoteEmail` for emails (not `\email` right away because it might conflict with RevTeX. Provide `\email` later after parsing attribs):

```
    \RequirePackage{url}%
    \DeclareUrlCommand\phfnote@format@url{}%
    \let\phfnoteEmail\phfnote@email
```

Set up everything according to the user’s selected attributes.

```
    \phfnote@internal@execattribs{phfnote@do@hyperrefdefs@attr@}{%
      phfnote hyperref-related option}{#1}%
```

And now, load the hyperref package (or don’t, if it’s deferred), and set some options.

```
    \phfnote@hyperrefdefs@loadhyperref
    \expandafter\hypersetup\expandafter{\phfnote@hyperrefdefs@val@options}
    \phfnote@hyperrefdefs@provideemail
    \phfnote@hyperrefdefs@redefeqref
    \urlstyle{notesf}
    \phfnote@hyperrefdefs@afterhook
  }
}
```

\phfnotePdfLinkColor

Set links color. Use as \phfnotePdfLinkColor{⟨*color*⟩}. Color may be any color name or specification recognized by the xcolor package.

```
\newcommand{\phfnotePdfLinkColor}[1]{%
  \@ifpackageloaded{xcolor}{%
    \colorlet{docnotelinkcolor}{#1}%
  }{% else:
    \PackageError{phfnote}{\protect\phfnotePdfLinkColor may only be
      used if the package xcolor is loaded.}{}%
  }%
}
```

\phfnote@sanitize@url
\phfnote@format@url
\phfnote@email

Provide base macros to be able to build up \email command for emails and other URL-like commands which should sanitize their arguments.

Also prepare the command \phfnoteEmail which will be renamed \email in our hyperref package setup (see above).

NOTE: The commands \phfnote@email and \phfnote@format@url will only work if you don’t have hyperrefdefs=false. They will work with hyperrefdefs=noload if you don’t want to load the hyperref package.

```
\def\phfnote@sanitize@url{%
  \catcode`\$12%
  \catcode`\&12%
  \catcode`\#12%
  \catcode`\^12%
  \catcode`\_12%
  \catcode`\%12%
  % \catcode`\^^J10%  newline = space
  % \catcode`\^^M10%  newline = space
  \relax%
}%
```

```
\def\phfnote@email{\begingroup\phfnote@sanitize@url\phfnote@impl@email@}%
\def\phfnote@impl@email@#1{\endgroup\href{mailto:#1}{\phfnote@format@url{#1}}}%
```

\phfnote@requirecolorpackage And finally define an internal utility to make sure that a color package (either color or xcolor) is loaded. If none are loaded, the xcolor package is loaded.

```
\def\phfnote@requirecolorpackage{%
  \@ifpackageloaded{color}{%
  }{%
    \@ifpackageloaded{xcolor}{%
    }{%
      \RequirePackage{xcolor}%
    }%
  }%
}
```

5.9 Cosmetic Font Definitions

\phfnote@do@fontdefs Minimalist cosmetic definition for fonts: load the T1 font encoding which is better. Also, use Computer Modern Bright as sans-serif font by default instead of Computer Modern Sans Serif.

If on XeTEX or LuaTEX, don't do anything.

```
\def\phfnote@do@fontdefs#1{
  \ifstrequal{#1}{false}{}{%
    \let\phfnote@tmp@do\@firstofone
    \IfFileExists{iftex.sty}{%
      \RequirePackage{iftex}%
      \phfnote@internal@setifxeorlua\phfnote@tmp@do\@gobble
    }{}
    \phfnote@tmp@do{
      \PassOptionsToPackage{T1}{fontenc}
      \RequirePackage{fontenc}
      \renewcommand\sfdefault{cmbr}
    }
  }
}
```

5.10 Bibliography Stuff

Provide some fixes for the bibliography.

\phfnote@bibstyle
\phfnote@bibfont
Our default bibliography style is stored in \phfnote@bibstyle. By default, it's our own hacked version of the naturemag style. The font in which to typeset the bibliography is stored in \phfnote@bibfont. By default, it's a little smaller than the main text.

```
\newcommand{\phfnote@bibstyle}{naturemagdoi}
\newcommand{\phfnote@bibfont}{\fontsize{9}{11}\selectfont}
```

\phfnote@bibliography These are a tentative implementation for \bibliography. The latter will be set to this implementation according to the user's package options.

```
\let\phfnote@old@bibliography\bibliography
\let\phfnote@old@bibliographystyle\bibliographystyle
\newcommand{\phfnote@bibliography}[1]{%
  \begingroup%
    \phfnote@bibfont%
    \phfnote@old@bibliographystyle{\phfnote@bibstyle}%
```

Our hack: make sure that the next instance of \section* will generate a TOC entry. (See \phfnoteHackSectionStarWithTOC.)

```
    \phfnoteHackSectionStarWithTOC%
```

Some special chars may appear in output of some ill-advised bibliography managers. Mostly the & symbol, such as in Taylor & Francis. We won't be needing a LaTeX alignment operator here, so just make & a normal printable character ("other" catcode).

```
    \catcode`\&=12\relax% normal char
```

Adjust the appearance of e-prints. We assume e-prints refer to the arXiv; here we generate a hyperlink and format them better.

```
    \providecommand\eprint[2][]{\href{http://arxiv.org/abs/##2}{arXiv:##2}}
```

Fix for RevTeX styles that use \doibase with a newline following them —

```
    \providecommand\doibase{\phfnote@doibasefix}
```

Relay the call to the "old" \bibliography command to actually implement the bibliography.

```
    \phfnote@old@bibliography{#1}%
  \endgroup%
}
\def\phfnote@doibasefix#110.{https://doi.org/10.}
```

\phfnote@bibliographystyle Tentative implementation of \bibliographystyle. Just register the new style in an internal variable, so that the style is actually loaded in \phfnote@bibliography.

This will be renamed to replace \bibliographystyle later, according to package options.

```
\newcommand{\phfnote@bibliographystyle}[1]{%
  \renewcommand{\phfnote@bibstyle}{#1}%
}
```

\phfnote@do@bibliographydefs Make our changes live. Will be called later according to package options.

```
\def\phfnote@do@bibliographydefs#1{%
  \ifstrequal{#1}{false}{}{%
    \let\bibliographystyle\phfnote@bibliographystyle%
    \let\bibliography\phfnote@bibliography%
  }
}
```

5.11 Better Footnote Style

\phfnote@do@footnotedefs Adjust the formatting of footnotes so they look better. Again, this is called later according to the package options.

```
\def\phfnote@do@footnotedefs#1{
  \ifstrequal{#1}{false}{}{%
    \let\phfnote@orig@makefnmark\@makefnmark
%%    \def\@makefnmark{\hbox{\@textsuperscript{%
%%        \normalfont\tiny\fontseries{sb}\selectfont\@thefnmark}}}
    \def\@makefnmark{\hbox{\@textsuperscript{%
          \normalfont\tiny\bfseries\@thefnmark}}}
%%    \def\@makefnmark{\hbox{\@textsuperscript{%
%%        \normalfont\scriptsize\bfseries\@thefnmark}}}% too large
  }
}
```

5.12 Other Stand-Alone Definitions and Helpers

5.12.1 A \notesmaller command

\notesmaller Relative font size command. Makes the text a fraction smaller than its surroundings. The fraction is either given explicitly as optional argument (1.0=same size) or is by default set by \notesmallerfrac.

To impalement this, we exploit the fact that LaTeX saves the current font size in the macro \f@size.

```
\newcommand\notesmaller[1][\notesmallerfrac]{%
  \fontsize{#1\dimexpr\f@size pt\relax}{#1\dimexpr\f@baselineskip pt\relax}%
  \selectfont\ignorespaces%
}
```

\notesmallerfrac Default fraction by which \notesmaller acts. Redefine to change defaults.

```
\def\notesmallerfrac{0.9}
```

5.12.2 Customized, “Inline,” Table of Contents

\inlinetoc

Just a customized table of contents. Horizontal rules before and after, and spacing is adjusted, and no “Contents” title. The table of contents looks just like at the top of this document. The command is described in subsection 4.3.

We call `\@starttoc` directly, bypassing the `\section*` included by `\tableofcontents` (see definition `\tableofcontents` in latex sources).

```
\newcommand{\inlinetoc}{%
  \begingroup%
    \vspace*{2mm}%
    \hrule%
    \vspace*{2mm}%
    \parskip=1pt\relax%
    \@starttoc{toc}%
    \vspace*{4mm}%
    \hrule%
    \vspace*{6mm}%
  \endgroup%
}
```

5.12.3 Inline commenting in documents

The code that was initially here was moved into a separate package: phfcc.

5.12.4 URL Styles

\url@notettstyle
\url@notesfstyle
\url@notesfssstyle
\url@noteitsfstyle
\url@notermstyle
\url@noteitstyle
\url@notesmlstyle

We also provide some URL styles. These can directly set with `\urlstyle{`⟨*style-name*⟩`}`.

```
\def\url@notettstyle{%
  \def\UrlFont{\ttfamily\notesmaller}%
  \phfnote@urlstyle@common%
}
\def\url@notesfstyle{%
  \def\UrlFont{\sffamily\notesmaller}%
  \phfnote@urlstyle@common%
}
\def\url@notesfssstyle{%
  \def\UrlFont{\fontfamily{cmss}\selectfont\notesmaller}%
  \phfnote@urlstyle@common%
}
\def\url@noteitsfstyle{%
  \def\UrlFont{\sffamily\itshape\notesmaller}%
  \phfnote@urlstyle@common%
}
\def\url@notermstyle{%
  \def\UrlFont{\rmfamily\notesmaller}%
  \phfnote@urlstyle@common%
}
\def\url@noteitstyle{%
  \def\UrlFont{\itshape\notesmaller}%
  \phfnote@urlstyle@common%
}
\def\url@notesmlstyle{%
  \def\UrlFont{\notesmaller}%
  \phfnote@urlstyle@common%
}
```

\phfnote@urlstyle@common The following code is common to all our styles. We do an ugly hack in which the tilde character (‘~’) is fixed to the tilde char in the Adobe Times font (ptm code), so that it looks nicer and its alignment is correct.

```
\def\phfnote@url@tilde{\hbox{\fontfamily{ptm}\selectfont\textasciitilde}}
%%\def\phfnote@url@tilde{\raise-0.8ex\hbox{%
%%    \kern-0.2ex\fontfamily{cmbr}\selectfont\textasciitilde}}
\def\phfnote@urlstyle@common{%
  \def\UrlTildeSpecial{\do\~{\phfnote@url@tilde}}%
  \let\Url@force@Tilde\UrlTildeSpecial%
}
```

5.12.5 Utility to Add TOC Entry For Starred Section

Here we provide an ugly hack which introduces an entry in the table of contents for \section* commands.

[Note: An existing way of adding the toc entry in these cases is to issue a \addcontentsline command before the relevant command (say \bibliography). However this is unreliable, because on page boundaries the \addcontentsline will pick up the previous page. This is why \addcontentsline should be issued right *after* the \section* command.]

WARNING

This command is truly a hack, don’t apply it globally! It forces (locally) the \section command to be followed by a ‘*’ ! Do this within a group, just before a command which you are sure is invoking \section* (such as \bibliography in the article class).

\phfnoteHackSectionStarWithTOC Locally force \section to be followed by * and introduce an entry in the table of contents.

```
\def\phfnoteHackSectionStarWithTOC{%
    \let\phfnote@old@section\section%
    \def\section*##1{\phfnote@old@section*{##1}\addcontentsline{toc}{section}{##1}}%
}
```

\phfnoteHackSectionStarWithTOCInCommand

Patches the given command (`#1`), which is known to invoke `\section*`, to locally first invoke `\phfnoteHackSectionStarWithTOC` and thus generate a TOC entry.

```
\def\phfnoteHackSectionStarWithTOCInCommand#1{%
  \expandafter\let\csname phfnote@old@\string#1\endcsname#1%
  \gdef#1{%
    \begingroup%
    \phfnoteHackSectionStarWithTOC%
    \csname phfnote@old@\string#1\endcsname%
    \endgroup%
  }%
}
```

5.12.6 Hack to save & restore a set of commands

Exactly what it sounds like. You can store a set of commands, specified by their name, by specifying an identifier. The commands corresponding to a given identifier can then later be restored.

\phfnoteSaveDefs

The command `\phfnoteSaveDefs{`⟨*identifier*⟩`}{`⟨*list of macro names*⟩`}` saves the current definitions of the given list of macro and associates them to the given identifier. The list of macros is specified as a comma-separated list of macro names.

```
\def\phfnoteSaveDefs#1#2{%
```

The macro `\phfnote@restoredefs@<identifier>` will store the code necessary to restore the macros.

```
  \csgdef{phfnote@restoredefs@#1}{}%
```

Iterate over the macros we are supposed to store.

```
  \def\@tmpa{#2}%
  \@for\next:=\@tmpa\do{%
```

For each macro we are supposed to store (whose name is given in `\next`), we `\let \phfnote@restoredefs@<identifier>@<macro-name>` store the current value of the macro.

```
    \global\csletcs{phfnote@restoredefs@#1@\next}{\next}%
```

Then, we append to `\phfnote@restoredefs@<identifier>` the code necessary to restore this macro. That code is simply a `\cslet` instruction.

Recall that `\xappto` expands its second argument (as `\xdef` does), allowing us to expand the value of `\next`.

```
    \expandafter\xappto\csname phfnote@restoredefs@#1\endcsname{%
      \noexpand\csletcs{\next}{phfnote@restoredefs@#1@\next}%
    }%
  }%
}
```

\phfnoteRestoreDefs Restores the macro saved by \phfnoteSaveDefs. We simply execute the macro \phfnote@restoredefs@<identifier>, in which we duly stored the code necessary to restore all the saved macros.

```
\def\phfnoteRestoreDefs#1{%
  \ifcsname phfnote@restoredefs@#1\endcsname%
    \csname phfnote@restoredefs@#1\endcsname%
  \else%
    \PackageError{phfnote}{\string\phfnoteRestoreDefs: no such
      definitions stored (#1)}{}
  \fi%
}
```

5.12.7 A utility for verbatim stuff in arguments of other macros

FIXME: DOCUMENT ME!

A utility for using verbatim stuff in arguments of other macros—exploit \detokenize

```
\def\phfverb#1{%
  \ifx\protect\relax%
    \phfverbfmt{\detokenize{#1}\unskip}%
  \else%
    \noexpand\phfverb{\unexpanded{#1}}%
  \fi%
}
\def\phfverbfmt#1{{\normalfont\texttt{#1}}}
```

5.13 Handle Package Options

5.13.1 Define and Parse Package Options

Initialization code for kvoptions for our package options. See section 3.

```
\SetupKeyvalOptions{
  family=phfnote,
  prefix=phfnote@opt@
}
```

[title=...] The title style to use. See subsubsection 4.1.1.

```
\DeclareStringOption[default]{title}
\DeclareVoidOption{notitle}{\def\phfnote@opt@title{false}}
```

[abstract=...] Option for abstract attributes (subsection 4.2).

```
\DeclareStringOption[]{abstract}
\DeclareVoidOption{noabstract}{\def\phfnote@opt@abstract{false}}
```

[pkgset=...] Option for Package sets (subsection 4.4)

```
\DeclareStringOption[rich]{pkgset}
```

[pagegeomdefs=...]
[pagegeom=...] Define the page geometry. See subsection 4.5.

```
\DeclareStringOption[default]{pagegeom}
\DeclareVoidOption{nopagegeom}{\def\phfnote@opt@pagegeom{false}}
```

Obsolete options—

```
\DeclareBoolOption[true]{pagegeomdefs}
\DeclareComplementaryOption{nopagegeomdefs}{pagegeomdefs}
```

—we handle these as follows. By default, `\ifphfnote@opt@pagegeomdefs` is true. If it isn't, that means it was overridden and we need to respect that.

[secfmt=...] Styling of section headings. See subsection 4.6.

```
\DeclareStringOption[section]{secfmt}
\DeclareVoidOption{nosecfmt}{\def\phfnote@opt@secfmt{false}}
```

[par=...] How to treat paragraphs. See subsection 4.7.

```
\DeclareStringOption[skip]{par}
\DeclareVoidOption{nopar}{\def\phfnote@opt@par{false}}
```

[spacingdefs=...] Add definitions to adjust spacing of lines and words. See subsection 4.8.

```
\DeclareStringOption[true]{spacingdefs}[true]
\DeclareVoidOption{nospacingdefs}{\def\phfnote@opt@spacingdefs{false}}
```

[fontdefs=...] Do some adjustments to the fonts. See subsection 4.9.

```
\DeclareStringOption[true]{fontdefs}[true]
\DeclareVoidOption{nofontdefs}{\def\phfnote@opt@fontdefs{false}}
```

[footnotedefs=...] Adjustments for footnotes. See subsection 4.10.

```
\DeclareStringOption[true]{footnotedefs}[true]
\DeclareVoidOption{nofootnotedefs}{\def\phfnote@opt@footnotedefs{false}}
```

[hyperrefdefs=...] Load hyperref and corresponding definitions. See subsection 4.11.

```
\DeclareStringOption[]{hyperrefdefs}[]
\DeclareVoidOption{nohyperrefdefs}{\def\phfnote@opt@hyperrefdefs{false}}
```

[bibliographydefs=...] Adjustments for bibliography, including default style. See subsection 4.12.

```
\DeclareStringOption[true]{bibliographydefs}[true]
\DeclareVoidOption{nobibliographydefs}{\def\phfnote@opt@bibliographydefs{false}}
```

\phfnote@loadpreset A helper macro to load presets. Can be used by presets that want to extend other presets.

```
\def\phfnote@loadpreset#1{%
  \IfFileExists{phfnotepreset-#1.def}{%
    \input{phfnotepreset-#1.def}%
  }{%
    \ifcsname phfnote@preset@#1\endcsname%
      \csname phfnote@preset@#1\endcsname%
    \else%
      \PackageError{phfnote}{Unknown preset: ‘#1’!}{You specified the
        option ’preset=...’ with an invalid value.  Please look up the
        package documentation corresponding to your version of phfnote
        for possible values.  Additionally, no file named ‘phfnotepreset-#1.def’
        was found.}%
  \fi%
  }%
}
```

[preset=...] Preset option. See subsection 2.2.

```
\define@key{phfnote}{preset}{%
  \phfnote@loadpreset{#1}%
}
```

Provide the standard error message for unknown options.

```
\DeclareDefaultOption{%
  \@unknownoptionerror
}
```

Small utility to deal with obsolete `XXXdefs=true/false` options. If the (obsolete) bool option #1 is set to false (which means it was set explicitly, so this must be respected), then emit a package warning and set option #2 (the regular option) to the string value `false`.

```
\def\phfnote@ifpkgoptfalsesetfalse#1#2{%
  \edef\x{%
    \expandafter\noexpand\csname ifphfnote@opt@#1\endcsname\noexpand\else
      \noexpand\PackageWarning{phfnote}{Option #1 is obsolete. Please use "#2=false’’ instead
      \noexpand\csgdef{phfnote@opt@#2}{false}\noexpand\fi}%
  \x
}
```

5.13.2 Define Global Presets

Define the global presets here. See subsection 2.2 for a description of what these presets do.

Some of the presets whose definitions are short are defined here directly here, in the sty file. Other presets are placed in a separate phfnotepreset-XXX.def file to avoid bloating the main style file.

Changed in v4.0 [2021/10/08]: Moved some presets to external .def files.

\phfnote@hook@atendload A hook for presets to do stuff at the end of package load.

```
\def\phfnote@hook@atendload{}
```

\phfnote@preset@article Article preset.

```
\def\phfnote@preset@article{
  \def\phfnote@opt@title{article}
  \def\phfnote@opt@par{indent}
  \def\phfnote@opt@pagegeom{default}
}
```

\phfnote@presetcommon@xnote Specify some common definitions for all our *note preset styles. The optional argument is the URL style to set.

```
\newcommand\phfnote@presetcommon@xnote[1][noteitsf]{
  \def\phfnote@opt@title{default}
  \def\phfnote@opt@par{skip}
  %\phfnote@opt@pagegeomdefstrue
  \def\phfnote@opt@pagegeom{wide}
  \setlength{\footnotesep}{5pt}
  \g@addto@macro\phfnote@hook@atendload{
    \ifdefined\urlstyle
      \urlstyle{#1}
    \fi
  }
}
```

\phfnote@preset@sfnote
\phfnote@preset@sfssnote
\phfnote@preset@opensansnote
\phfnote@preset@utopianote
\phfnote@preset@mnmynote

Define the different *note styles.

```
\def\phfnote@preset@sfnote{
  \phfnote@presetcommon@xnote
  \def\phfnote@opt@footnotedefs{true}
  \def\phfnote@opt@fontdefs{true}
  \renewcommand\familydefault{\sfdefault}
  \renewcommand{\notesectionallfontfamily}{\sfdefault}
}
\def\phfnote@preset@sfssnote{
```

set up all the settings as for `sfnote` …

```
  \phfnote@loadpreset{sfnote}%
```

… but override:

```
  \def\phfnote@opt@fontdefs{false}
  \PassOptionsToPackage{T1}{fontenc}
  \RequirePackage{fontenc}
  \renewcommand\sfdefault{cmss}
}
\def\phfnote@preset@opensansnote{
```

set up all the settings as for `sfnote` …

```
  \phfnote@loadpreset{sfnote}%
```

… but override:

```
  \def\phfnote@opt@fontdefs{false}
  \PassOptionsToPackage{T1}{fontenc}
  \RequirePackage{fontenc}
  \PassOptionsToPackage{default,scale=0.9}{opensans}
  \RequirePackage{opensans}
}
\def\phfnote@preset@utopianote{
  \phfnote@presetcommon@xnote[noteit]
  \def\phfnote@opt@fontdefs{false}
  \PassOptionsToPackage{T1}{fontenc}
  \RequirePackage{fontenc}
  \RequirePackage{fourier}
  \renewcommand{\notesectionallfontfamily}{put}
  \renewcommand{\notetitlefont}{\bfseries}
  \renewcommand{\sfdefault}{phv}
}
\def\phfnote@preset@mnmynote{
  \phfnote@presetcommon@xnote[noteit]
  \def\phfnote@opt@footnotedefs{false}
  \def\phfnote@opt@fontdefs{false}
  \PassOptionsToPackage{T1}{fontenc}
  \RequirePackage{fontenc}
  \renewcommand{\notesectionallfontfamily}{\sfdefault}
```

Require these packages AFTER the default package set, because some symbols may be defined in package sets, and I've had problems with re-definitions etc… anyway this seems to work this way:

```
  \g@addto@macro\phfnote@hook@atendload{
    \RequirePackage{MnSymbol}
    \PassOptionsToPackage{medfamily,textosf,mathlf,minionint,footnotefigures}{MinionPro}
    \RequirePackage{MinionPro}
    \PassOptionsToPackage{medfamily}{MyriadPro}
    \RequirePackage{MyriadPro}
  }
}
```

\phfnote@preset@pkgdoc Preset for a package documentation.

Start by setting the same settings as for other Xnote presets.

```
\def\phfnote@preset@pkgdoc{
  \phfnote@presetcommon@xnote[noteit]
  \def\phfnote@opt@fontdefs{false}
```

Then set up the font, which is done in a separate macro \phfnote@pkgdoc@setupfont in case individual documents would like more specific settings. (For example, some packages may want a different math font.)

```
  \phfnote@pkgdoc@setupfont
```

Finally, set up general appearance.

```
  \def\phfnote@opt@secfmt{section,paragraph,itpar,blockpar,larger,secsquares,secnummargin}
  \def\phfnote@opt@pagegeom{bigmargin}
  \def\phfnote@opt@abstract{noname}
}
```

Also provide a helper macro which is to load the font packages we want. By default, we use Utopia fonts via the fourier package, but some package documentations may want a different math font. Override \phfnote@pkgdoc@setupfont to adjust the whole font set-up, or \phfnote@pkgdoc@setupmainfont to adjust only the main document font.

Changed in v3.1 [2020/05/25]: Fixes for more recent versions of the opensans package..

```
\providecommand\phfnote@pkgdoc@setupfont{
  \PassOptionsToPackage{T1}{fontenc}
  \RequirePackage{fontenc}
  \phfnote@pkgdoc@setupmainfont
  \renewcommand{\notesectionallfontfamily}{put}
  \renewcommand{\notetitlefont}{\bfseries}
  \IfFileExists{opensans.sty}{}{\PackageError{phfnote}{Font OpenSans is not
      available (need ‘opensans’ package)}{Please install the opensans
      package, which provides the OpenSans font.}}
  \PassOptionsToPackage{scale=0.85,defaultsans}{opensans}
  \RequirePackage{opensans}
}
\providecommand\phfnote@pkgdoc@setupmainfont{\RequirePackage{fourier}}
```

\phfnote@preset@reset Finally, the reset preset:

```
\def\phfnote@preset@reset{
  \def\phfnote@opt@pkgset{none}
  \def\phfnote@opt@title{false}
  \def\phfnote@opt@pagegeom{false}
  \def\phfnote@opt@spacingdefs{false}
  \def\phfnote@opt@par{false}
  \def\phfnote@opt@abstract{false}
  \def\phfnote@opt@hyperrefdefs{false}
  \def\phfnote@opt@fontdefs{false}
  \def\phfnote@opt@secfmt{false}
  \def\phfnote@opt@bibliographydefs{false}
  \def\phfnote@opt@footnotedefs{false}
```

> **WARNING**
>
> NOTE TO SELF: DO NOT FORGET TO ADD HERE RESET COMMANDS FOR ANY NEW OPTION THAT WE PROVIDE IN THE FUTURE.

```
}
```

5.13.3 Finally, Process and Execute the Package Options

Process the options:

```
\ProcessKeyvalOptions*
```

Take action according to the user options.

For the `pkgset` option.

```
\expandafter\phfnote@do@pkgset\expandafter{\phfnote@opt@pkgset}
```

For the `title` option.

```
\expandafter\phfnote@do@notetitle\expandafter{\phfnote@opt@title}
```

For the `abstract` option.

```
\expandafter\phfnote@do@noteabstract\expandafter{\phfnote@opt@abstract}
```

For the `secfmt` option.

```
\expandafter\phfnote@do@secfmt\expandafter{\phfnote@opt@secfmt}
```

For the `pagegeomdefs` option. Here, the first line is needed to deal with the obsolete option '`pagegeom`'.

```
\phfnote@ifpkgoptfalsesetfalse{pagegeomdefs}{pagegeom}
\expandafter\phfnote@do@pagegeom\expandafter{\phfnote@opt@pagegeom}
```

For the spacingdefs option.

```
\expandafter\phfnote@do@spacingdefs\expandafter{\phfnote@opt@spacingdefs}
```

For the par option.

```
\expandafter\phfnote@do@par\expandafter{\phfnote@opt@par}
```

For the hyperrefdefs option.

```
\expandafter\phfnote@do@hyperrefdefs\expandafter{\phfnote@opt@hyperrefdefs}
```

For the fontdefs option.

```
\expandafter\phfnote@do@fontdefs\expandafter{\phfnote@opt@fontdefs}
```

For the bibliographydefs option.

```
\expandafter\phfnote@do@bibliographydefs\expandafter{\phfnote@opt@bibliographydefs}
```

For the footnotedefs option.

```
\expandafter\phfnote@do@footnotedefs\expandafter{\phfnote@opt@footnotedefs}
```

Finally, execute the hook we set up for definitions at the end of the package loading:

```
\phfnote@hook@atendload
```

5.14 Helper files

Now we write some code to some external .def files for large chunks of code that are only needed in specific situations, in order to avoid bloating the main style file.

5.14.1 The *xpkgdoc* preset

The xpkgdoc preset is based on the pkgdoc preset. It introduces multiple pretty involved tools and definitions on top of pkgdoc; these are not needed unless this preset is loaded. Let's write all of the corresponding definitions into a separate .def file.

```
<*phfnotepreset-xpkgdoc>
```

Load the pkgdoc preset before anything else:

```
\phfnote@loadpreset{pkgdoc}
```

Include the verbdef package, because it's always useful.

```
\RequirePackage{verbdef}
```

Some patching first: Patch up `\PrintChanges` and `\PrintIndex`, if they are defined (for if we are using the ltxdoc package for latex package documentation). We want these to generate an entry in the table of contents. Also provide the utility `\PrintChangesAndIndex`, which calls both `\PrintChanges` and `\PrintIndex` with some additional spacing.

```
\ifdefined\PrintChanges
  \phfnoteHackSectionStarWithTOCInCommand\PrintChanges
\fi
\ifdefined\PrintIndex
  \phfnoteHackSectionStarWithTOCInCommand\PrintIndex
\fi
\def\PrintChangesAndIndexSpacing{\vspace{3cm plus 2cm minus 2cm}}
\def\PrintChangesAndIndex{\PrintChangesAndIndexSpacing\PrintChanges
  \PrintChangesAndIndexSpacing\PrintIndex}
```

Set the index to TWO columns only (three is too tight).

```
\ifdefined\c@IndexColumns
  \setcounter{IndexColumns}{2}
\fi
```

And set the glossary, that is, the list of changes history to single-column. For this, renew the environment completely to remove the `multicols` environment.

```
\let\phfnote@xpkgdoc@old@theglossary\theglossary
\let\phfnote@xpkgdoc@old@endtheglossary\endtheglossary
\renewenvironment{theglossary}{%
  \glossary@prologue%
  \GlossaryParms \let\item\@idxitem \ignorespaces}
{}
```

Tools to condense the macro names in the margins, and make them break at the margin width (at arbitrary points in the macro name) instead of overflowing on the documentation text.

```
\def\phf@fillwithdiscretionaries#1#2#3{%
  \def\phf@fillwithdescretionaries@a{#1}%
  \def\phf@fillwithdescretionaries@b{#2}%
  \edef\x{#3}%
  \expandafter \phf@fillwithdiscretionaries@ \x \@nil
}
\def\phf@fillwithdiscretionaries@#1#2\@nil{%
  \if\relax\detokenize{#1}\relax\else
    #1%
    \if\relax\detokenize{#2}\relax\else
      \discretionary{\phf@fillwithdescretionaries@a}{%
```

```
        \phf@fillwithdescretionaries@b}{}%
      \phf@fillwithdiscretionaries@#2\@nil
    \fi
  \fi
}
\definecolor{phfxpkgdocmacronamehyphencolor}{rgb}{0.6,0.6,0.7}
\def\phf@xpkgdoc@macrobreakhyphen{%
  \hbox{\textcolor{phfxpkgdocmacronamehyphencolor}{-}}}
\def\ScaleHorizontallyAndHyphenateAnywhere#1#2{% {hscale-factor}{text}
  \scalebox{#1}[1.0]{%
    \parbox{% compute inverse of #1 ...
      \dimexpr \marginparwidth*65536 /
      \number\dimexpr #1\p@ \relax \relax
    }{\raggedleft\phf@fillwithdiscretionaries{%
        \phf@xpkgdoc@macrobreakhyphen}{}{#2}\hfill}%
  }%
}
\def\phf@xpkgdoc@marginmacronamecompressfactor{0.85}
\def\phf@xpkgdoc@macrofont{\small\bfseries}% \footnotesize ?
\def\PrintMarginLabelContents#1{%
  \strut\MacroFont\phf@xpkgdoc@macrofont
  \ScaleHorizontallyAndHyphenateAnywhere{%
    \phf@xpkgdoc@marginmacronamecompressfactor}{#1}%
  \par
}
\def\PrintMarginLabel#1{\marginpar{\PrintMarginLabelContents{#1}}}
\def\PrintDescribeMacro#1{\PrintMarginLabelContents{\string#1}}
\let\PrintMacroName\PrintDescribeMacro
\let\PrintDescribeEnv\PrintMarginLabelContents
\let\PrintEnvName\PrintMarginLabelContents
```

Hyperref: Request default hyperref definitions with `hyperrefdefs=defer`, but we'll load hyperdoc instead of hyperref.

```
\def\phfnote@opt@hyperrefdefs{defer}
\g@addto@macro\phfnote@hook@atendload{
  \RequirePackage{hypdoc}
  \urlstyle{noteit}
}
```

Provide Macro: \pkgname{⟨*package name*⟩} to format a package name. Also place it in the general index. This command is robust and can be used in section titles etc.

```
\def\pkgname#1{%
  \pkgnamefmt{#1}%
  \index{#1=\pkgnamefmt{#1}|hyperpage}%
  \index{packages:>#1=\pkgnamefmt{#1}|hyperpage}%
}
\robustify\pkgname
\def\pkgnamefmt#1{\textsf{#1}}
```

```
\robustify\pkgnamefmt
```

Provide Macros: \changed and \changedreftext, with more advanced support for displaying changes in package functionality or API.

First, we need a counter for the x-ref system.

```
\newcounter{phfnotechanged}
```

Mark changes in the implementation section of the package documentation with the command \changed[⟨*label name*⟩]{⟨*v1.0*⟩}{⟨*2016/05/22*⟩}{⟨*description*⟩}. This command automatically adds the change to the package's change history list, and allows you to refer to this change anywhere else in the package doc with \changedreftext.

```
\newcommand*\changed[4][]{%
```

First, if no label is given as optional argument, then just display the change and add it to the package changes list.

```
  \if\relax\detokenize{#1}\relax%
    \changedtextfmt{#2}{#3}{#4}%
    \changes{#2}{#3}{#4}%
  \else%
```

If a label name is provided as optional argument, then we need to write some stuff to the .aux file to make the change visible in the whole document.

```
    \protected@edef\phfnotechanged@tmpa{{#2}{#3}{#4}}%
    \immediate\write\@auxout{\string\phfnote@changed@set%
      {#1}{\expandonce\phfnotechanged@tmpa}}%
    \par\hspace*{0pt}\refstepcounter{phfnotechanged}\label{phfnotechanged:#1}%
    \begingroup\let\phfnote@changedreftext@par\relax
      \changedreftext[\@secondoftwo]{#1}%
    \endgroup
    \changes{#2}{#3}{\hyperref[phfnotechanged:#1]{#4}}%
  \fi
}
\def\phfnote@changed@set#1{%
  \expandafter\gdef\csname phfnote@changed@lbl@#1\endcsname%
}
```

When you document changes with the help of \changed, you may refer to any specific change from anywhere else in the package doc with the help of \changedreftext{⟨*label name*⟩}.

```
\def\phfnote@changedreftext@par{\par}
\newcommand*\changedreftext[2][\phfnote@changedrefto]{%
  \phfnote@changedreftext@par%
  \ifcsname phfnote@changed@lbl@#2\endcsname
    #1{#2}{%
      \expandafter\expandafter\expandafter\changedtextfmt%
```

```
          \csname phfnote@changed@lbl@#2\endcsname
    }
  \else
    \hyperref[phfnotechanged:#2]{%
      \changedtextfmt{???}{???}{[\textbf{missing ref}]}%
    }%
  \fi
  \par
}
\def\phfnote@changedrefto#1{\hyperref[phfnotechanged:#1]}
```

The macro \changedtextfmt{⟨*v1.0*⟩}{⟨*2016/05/22*⟩}{⟨*description*⟩} takes care of formatting the change on the spot.

```
\newcommand*\changedtextfmt[3]{%
    \textit{Changed in {#1\kern 0.3ex\relax[#2]}:} #3.
}
```

Provide environment pkgoptions**:** Set up an elaborate environment (based on a description environment) to describe package options.

```
\RequirePackage{enumitem}
\newlist{pkgoptions}{description}{1}
\setlist[pkgoptions]{font=\pkgoptionfmt@many[{\vspace*{5pt}}],style=nextline}
```

But patch the pkgoptions' \item command, so that it puts an additional pair of braces around its argument. In this way, the font= attribute for the list sees the full label as its next token, and can be used as a macro argument. (This is not needed for newer versions of enumitem.)

```
\def\pkgoptions@item{\@ifnextchar[\pkgoptions@item@\pkgoptions@item@@}%]
\def\pkgoptions@item@[#1]{\pkgoptions@old@item[{{#1}}]}%
\def\pkgoptions@item@@{\PackageWarning{phfnote}{{pkgoptions}: you must
    specify label to \string\item as \string\item[label].}%
  \pkgoptions@old@item}%
\apptocmd\pkgoptions{%
  \let\pkgoptions@old@item\item
  \let\item\pkgoptions@item
}{}{\PackageWarning{phfnote}{preset xpkgdoc: Failed to patch command
    \string\pkgoptions}}
```

For convenience, also provide a \meta-like command for boolean arguments (true or false). '\metatruefalsearg' typesets as '⟨true | false⟩'.

```
\def\metatruefalsearg{\meta{\phfverb{true} $\mid$ \phfverb{false}}}
```

Include also a command to format a package option. Puts the option in a box in typewriter text style, and indexes it. The optional argument is meant to be internal—it adds commands after the displayed text (use it to add, e.g. spacing).

When indexing the packages, make sure to remove the protective braces if any.

```
\newcommand\pkgoptionfmt[2][]{%
  \begingroup\let\meta\pkgoptfmt@meta\pkgopt@fbox{\normalfont\ttfamily #2}\endgroup%
  \expandafter\phfnote@pkgdoc@index\expandafter{\@firstofone #2}%
  #1}
\newcommand\pkgoptionfmt@many[2][]{%
  \def\pkgoptionfmt@tmp@addcomma{}
  \def\do##1{\pkgoptionfmt@tmp@addcomma\pkgoptionfmt{\vphantom{#2}##1}%
    \def\pkgoptionfmt@tmp@addcomma{\hskip0.25em\relax,\hskip0.8em\relax}}%
  \expandafter\docsvlist\expandafter{\@firstofone #2}%
  #1}
\let\pkgopt@save@meta\meta
\def\pkgopt@fbox{\fbox}
\def\pkgoptfmt@meta#1{\begingroup\normalfont\itshape\pkgopt@save@meta{#1}\endgroup}
```

Whenever a package option is formatted with `\pkgoptionfmt`, it is placed in the index. Because package options may be of the form `key=val`, we want to split keys from values and put them independently in the index. This is done by entering a TEX group, and using an `\lccode` trick: the code is prepared to iterate over a list of comma-separated stuff, but then the "lowercase" version of that code is executed instead, where the ='s have been replaced by `,`'s.

```
\def\phfnote@pkgdoc@index#1{%
  \begingroup\lccode`\= = `\,\relax%
    \def\x{\lowercase{\def\@tmpa{#1}}}%
    \x%
    \let\meta\@gobble%
    \let\marg\@gobble%
    \let\oarg\@gobble%
    \let\parg\@gobble%
    \let\vphantom\@gobble%
    \let\hphantom\@gobble%
    \let\pkgoptattrib\@firstofone%
    \let\pkgoptattribnodots\@firstofone%
    \let\pkgoptattribempty\@empty%
    \def\handleitemindex##1{%
      \edef\@tmpc{##1}%
      \if\relax\detokenize\expandafter{\@tmpc}\relax\else%
        \edef\@tmpb{{\expandonce\@tmpc=\string\verb!*+\expandonce\@tmpc+ (\pkgoptname)|hyperp
        \expandafter\index\@tmpb%
        \edef\@tmpb{{\packageoptionsname:>\expandonce\@tmpc=\string\verb!*+\expandonce\@tmpc+
        \expandafter\index\@tmpb%
      \fi%
    }%
    \def\@tmpc{\forcsvlist{\handleitemindex}}%
    \expandafter\@tmpc\expandafter{\@tmpa}%
  \endgroup%
}
\def\pkgoptname{pkg. opt.}
\def\packageoptionsname{package options}
```

Provide environment `cmdoptions`**:** hijack the `pkgoptions` environment to do the same thing, except we place the items in the index under “command options” instead of “package options.”

```
\def\cmdoptions{\begingroup\setcmdnotpkgoptions
  \pkgoptions}
\def\endcmdoptions{\endpkgoptions\endgroup}
\newcommand\cmdoptionfmt[2][]{\begingroup\setcmdnotpkgoptions
  \pkgoptionfmt[{#1}]{#2}\endgroup}
\def\cmdoptname{cmd. opt.}
\def\commandoptionsname{command options}
\def\setcmdnotpkgoptions{\let\pkgoptname\cmdoptname
  \let\packageoptionsname\commandoptionsname
  \let\pkgopt@fbox\cmdoptionsfbox}
\def\cmdoptionsfbox#1{\ensuremath{\underline{{\text{#1}}}}}
```

Provide the `\pkgoptattrib` command, which typesets its argument as `\{arg, ...\}`—useful to typeset attributes such as in subsection 4.2. The variant `\pkgoptattribondots{arg}` typesets `\{arg\}` while `\pkgoptattribempty` expands to `{}`.

```
\def\pkgoptattrib#1{\{#1,...\}}
\def\pkgoptattribnodots#1{\{#1\}}
\def\pkgoptattribempty{\{\}}
```

Colorful boxes: environments `pkgnote`**,** `pkgwarning`**, and** `pkgtip`**.** Now, load the tcolorbox package to provide visual “Note,” “Warning,” and “Tip” boxes. Because tcolorbox includes the verbatim package which messes up the `verbatim` environment in latex dtx files (for which source lines all start with a `%` which needs to be stripped), we save the `verbatim`-related commands, and restore them after the interfering packages have been loaded.

```
\phfnoteSaveDefs{verbatimstuff}{%
  verbatim,@verbatim,@xverbatim,@sxverbatim,endverbatim}
\usepackage{tcolorbox}
\newtcolorbox{pkgnote}{
  colback=blue!5!white,
  colframe=blue!5!white,
  coltitle=blue!50!black,
  toptitle=1.5ex,
  fonttitle=\bfseries,
  title={NOTE}
}
\newtcolorbox{pkgwarning}{
  colback=red!5!white,
  colframe=red!5!white,
  coltitle=red!50!black,
  toptitle=1.5ex,
  fonttitle=\bfseries,
  title={WARNING}
}
\newtcolorbox{pkgtip}{
  colback=green!5!white,
  colframe=green!5!white,
  coltitle=green!50!black,
  toptitle=1.5ex,
  fonttitle=\bfseries,
  title={TIP}
}
\phfnoteRestoreDefs{verbatimstuff}
```

Patch the `verbatim` environment to remove extraneous space after the environment caused by I don't know what weird cause:

```
\appto\endverbatim{\vspace{-\baselineskip}}
```

Common title stuff:

```
\def\phfqitltxPkgTitle#1{The \pkgname{#1} package\thanks{\itshape
  This document corresponds to \pkgname{#1}~\fileversion, dated \filedate. It
  is part of the
  \href{https://github.com/phfaist/phfqitltx/}{\pkgname{phfqitltx}} package
  suite, see \url{https://github.com/phfaist/phfqitltx}.}}
```

Utility to parse package file date into "`\today`"-style date: invoke as `\date{\pkgfmtdate\filedate}`.

```
\def\pkgfmtdate#1{%
  \edef\pkgfmtdate@thedate{#1}%
  \expandafter\pkgfmtdate@next\pkgfmtdate@thedate\@nil%
}
\def\pkgfmtdate@next#1/#2/#3\@nil{% YYYY/MM/DD
  \ifcase #2 \or January\or February\or March\or April\or May%
  \or June\or July\or August\or September\or October%
  \or November\or December\fi\space #3,%
  \space #1}
\robustify\pkgfmtdate@next
```

`</phfnotepreset-xpkgdoc>`

Change History

v1.0

- General: Initial version 1

v1.1

- General: If running XeTeX or LuaTeX, then do not load inputenc and fontenc as part of `rich` and `extended` package sets. Plus, do not load inputenc (resp. fontenc) if the package is already loaded 15

v3.0

- General: Added support for inline commenting using `\phfMakeCommentingCommand` 24
- Improved `abstract,noabstract` package options syntax 14
- Changed how to turn on/off the page geometry settings by improving the `pagegeom` package options, deprecated `pagegeomdefs` option 16
- Changed the `xwide` page geometry for tighter vertical margin for single-column text (use `pagegeom=xwidev1` instead for old behavior) 16
- Improved `par,nopar` package options syntax 20
- Improved `secfmt,nosecfmt` package options syntax 19
- Added the `pretty` and `pretty2` title styles 8
- The `article` title style was redesigned to use our new title engine, with improved spacing of the elements in all cases including multiline titles, author and/or date not specified, and presence or absence of thanks notes. Use `title=articlev1` for old behavior 9
- The default title style was redesigned to improved spacing of the elements in all cases including multiline titles, author and/or date not specified, and presence or absence of thanks notes. Use `title=defaultv1` for old behavior 7
- Improved `title,notitle` package options syntax 9

v3.1

- `\phfnote@preset@pkgdoc`: Fixes for more recent versions of the opensans package. 66
- General: Moved support for inline commenting to the separate dedicated package phfcc 24

v3.2

- General: We now redefine `\eqref` by default to include the parentheses inside the hyperlink. Use the `hyperrefdefs=noeqref` package option to disable this feature 22

v4.0

- General: Moved some presets to external `.def` files 64

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.

</package>