The *standalone* Class and Package

Martin Scharrer

martin@scharrer-online.de

http://www.ctan.org/pkg/standalone/

Version v0.3a – 2010/03/27

1 Introduction

Larger LaTeX documents can be split into multiple TeX files which are then included in a main document with `\include` for e.g. chapter files or `\input` for e.g. TeX-coded pictures. Keeping pictures in their own sub-files improves readability of the main file and simplifies the sharing of them between different documents. However, during the, sometimes lengthly, drawing/coding process it has benefits to be able to compile the pictures on their own. The compile process is much quicker and the resulting document only holds the picture which avoids constant page turning and zooming.

While it is possible to write a small 'main' file for each picture file, this method is a little cumbersome and clutters the directories with a lot of extra files. A second method is to place the 'main' components, i.e. a preamble, directly into the picture files and make the main document ignore this code sections.

The package `standalone` can be used in the main document to skip all extra preambles in included files. The main file must load all packages and settings required by the sub-files. Several package options are provided to collect the preambles of the sub-files automatically and execute them from the main file.

A `standalone` class is also provided to minimise the extra preamble code needed in this files. It's usage is optional, but simplifies and standardises how picture files are compiled standalone. The class uses by default the `preview` package to create an output file which only contains the picture with no extra margins, page numbers or anything else. A configuration file `standalone.cfg` read by the class allows the user to adjust settings and macros easily on a per directory base.

Similar Packages

The `docmute` package by T.M. Trzeciak is written for the same basic task as the `standalone` package. However, no sub-preamble processing other than the removal is support. It also doesn't provide a special class or configuration file.

2 Usage

2.1 Quick instructions

Load the `standalone` *package* very early in the main document. Also all packages needed by all the sub-files must be loaded by the main document. Include your picture or other sub-files using `\input` or a similar macro as normal. In the sub-files use the `standalone` *class* with a normal `\documentclass` and load all packages needed for the particular file. Finally wrap the actual content of the sub-file in a `document` environment.

When the sub-file is compiled on its own the `\documentclass` and `document` environment will be active as normal. The main file, however, will skip everything from the `\documentclass` till the `\begin{document}`. The (now fake) `document` environment is redefined to be a simple TeX-group. Any code after the `\end{document}` will be ignored. The real `document` environment of the main file will be unaffected and will work as normal.

Instead of transferring the packages required by each sub-file manually to the main document preamble, this task can be automatised using the options listed in section 2.3

2.2 Class Options

The `standalone` class will load a real document class. By default this is `article`. The document class normally has not much influence on sub-files like pictures, especially when the `preview` package is active. However, the used class can be adjusted by the user with the `class=`⟨*class name*⟩ option.

A special `beamer` option is provided to handle beamer frames and overlays correctly. See section 2.7 for more information.

All other used options are passed to the loaded class.

2.3 Package Options

subpreambles

The `standalone` package removes all sub-file preambles ("sub-preambles") by default when loaded. However, if the package is loaded with the subpreambles options, all sub-preambles are stored in an auxiliary file with the name '⟨*main tex file name*⟩`.sta`' (for *sta*ndalone). This file is then loaded or processed at the beginning of the next LaTeX run (i.e. at the place in the preamble where the `standalone` package is loaded). The way how the subpreambles option works can be controlled by the options sort, print and comments/nocomments. Please note that the sort and print options require of course the subpreambles option and will enable it if not already done so.

With only the subpreambles option set, the sub-preambles are simple read and executed unchanged. This includes the risk of option clashes if one package is loaded with different options inside the sub-preambles and/or the main preamble.

sort

This is avoided by the sort option, which accumulates all packages loaded by all sub-files together with their options. The options are then marked to be loaded

by the package using LaTeXs `\PassOptionsToPackage` macro. The packages are loaded at the end of the preamble using the `\AtBeginDocument` hook. This allows the user to load the same packages with own options in the main file, after the `subversion` package is loaded, without any option clashes.

While the sort option is giving already good results, problems with the order of packages can still occur. Some packages provide, redefine or patch the same macros, so that they must be loaded in the correct order to give the desired result. Potential additional code in the sub-preambles, required for some sub-figures but maybe incompatible with others, complicates the situation further. If such issues occur they can hardly be handled in an automatic way. Instead the sub-preambles must be carefully merged into the main preamble. The option print was created to simplify this otherwise cumbersome task. It concatenated all sub-preambles into a single file named '⟨*main tex file name*⟩`.stp`' (for *st*andalone, *p*rint). Each preamble is commented with its original file name. Please note that `.sta` file mentioned above, while quite similar, holds additional macros and might not be easily user readable or editable. After the file was generated it can be easily pasted into the main file preamble using a text editor.

print

When the print option is enabled the normal `.sta` file is not generated or loaded. Because this will cause most likely some errors related to packages not loaded, all sub-file bodies will be skipped. A warning is printed for each sub-file to remind the user about this fact. The print option is only indented to by used when required to get a list of sub-preambles. After including this list in the main file the option must be removed to compile the main file normally.

print, sort

Finally if both the print and sort options are enabled, a 'sorted' list of sub-preambles is printed into the `.stp` file. In this 'sorted print' mode all `\usepackage` macros (and similar macros like `\usepgflibrary`, `\usetikzlibrary` and `\usetikztiminglibrary` from the `pgf`, `tikz` and `tikz-timing` packages, respectively) are removed from the rest of the sub-preamble code. A list of packages (and libraries) without duplicates is printed at the begin of the `.stp` file (using `\usepackage`, of course). Every option provided by any sub-file for a package is added, again without duplicates. If specific package date was requested in a sub-file it is also added. If multiple dates are requested for one package, the most recent (i.e. the "highest one", not the last processed) is used. After this list(s) the rest of the sub-preamble code is printed with the above macros removed. This mode frees the user from the need to remove duplicates and collect package options manually.

Please note that all `\usepackage` and similar macros inside braces `{ }` will not be seen by `standalone`s sort macro and therefore are not extracted or handled in any special form mentioned above. This can be exploited to load certain packages only in `standalone` mode but not in the main document. Unfortunately, macros inside `\ifstandalone`...`\fi` are seen and extracted while not wanted inside the main file. The macro `\onlyifstandalone{`⟨*code*⟩`}` (see below) was created because of this two reasons. Its argument braces hide the content from the scanner. It is then also completely removed from the printed sub-preamble code.

comments/nocomments

The complementary options comments/nocomments select if the `.stp` file should also include the comments of the sub-preambles. Comments are included

by default in the non-sorting print mode (print without sort option), but can cause 'wrong' results during the 'sorting' process and are therefore removed by default in this mode. The reason for this can be explained as follows. In order to transfer the comments from the sub-files to the `.stp` file TEX must be instructed to handle them as normal input and not discard them. However, in this case the scanning algorithm which removes `\usepackage` and friends can not distinguish between 'active' macros and macros which are commented out. All above mentioned macro inside comments will then be processed as when there where 'active'. The user might favour the information provided by the comments over this small risk and enable them using the comments option.

2.4 Dependencies

The `standalone` class requires the `kvoptions` package (Oberdiek bundle) and the `preview` package. Both should be available in a standard LATEX installation. The `beamer` option of course requires the `beamer` bundle to be installed. The `standalone` package does not require other packages, but can take advantage from the `fink` package (*Fi*le *N*ame *K*eeper), to access the filenames of the sub-files. For this the `fink` package must be loaded before `standalone`. Without it a file name macro defined by LATEX itself is used instead which should do also fine, but is reset at every `\input` macro. Placing this macro before `\documentclass` without marking it with `\csname standaloneignore\endcsname` will then lead to wrong file names inside the `.sta` and `.stp` files.

2.5 Environments and Macros

`standalone` The `standalone` environment is wrapped around the content of each sub-file when compiled standalone. By default it only contains a `preview` environment as long the class is not called with the `preview=false` option. It can be redefined in the configuration file if required. When compiled as part of a main document the `standalone` environment does nothing (apart of being a TEX group).

`standaloneframe` The `standaloneframe` environment is only defined when the class is called with the `beamer` option and acts as a replacement of the `frame` environment of beamer when compiled standalone. All optional arguments of `frame` are supported. When compiled as part of a main document it does nothing except of gobbling its arguments.

`\ifstandalone` Both the class and the package provide the if-switch `\ifstandalone`, which can be used to only include code if the file is compiled standalone. The switch is set to `\iftrue` by the class and to `\iffalse` by the package.

The additional file `standalone.tex` also defines this switch by checking if `\documentclass` was already used. It can be included with `\input{standalone}` and is intended for specialised files which do not use the `standalone` class.

`\ifstandalonebeamer` Both the class and the package provide the if-switch `\ifstandalonebeamer`, which can be used to only include code if the file is compiled standalone with the

`beamer` class option set. The switch is set to `\iftrue` by the class when loaded with the `beamer` option and always to `\iffalse` by the package.

`\onlyifstandalone{`⟨*code*⟩`}` This is the macro version of the `\ifstandalone` if-switch. It executes ⟨*code*⟩ only in `standalone` mode. As mentioned in section 2.3 it can also be used to hide `\usepackage` and similar macros from the extraction scanner of the `sort` option. The macro and its argument is not printed into the `.stp` file.

`\standaloneignore` In rare cases some code must be placed before the `\documentclass` of a sub-file (e.g. `\PassOptionsToPackage`). Because the main document will only skip code between `\documentclass` and `\begin{document}` this code will be executed by it. In order to avoid this the macro `\standaloneignore` can be used at the very beginning of a sub-file to skip over this code. However it must be written as `\csname standaloneignore\endcsname` to avoid a 'Undefined control sequence' error when compiled standalone. After all the class is not loaded at this point, therefore no `standalone` macros are yet defined. The `\csname...\endcsname` construct will simple make it equal to `\relax` in this case.

Please note that all code before `\documentclass` is not processed by any of the subpreamble options but always simply removed. This macro was inspired by the similar macro `\docmute` of the `docmute` package.

2.6 Usage Examples

Example 1: Use of *standalone* package.

```
% Main file
% Real document class:
\documentclass{article}

% Use the 'standalone' package:
\usepackage{standalone}

% Load all packages needed for all sub-files:
\usepackage{tikz}

% Inside the real 'document' environment
% read the sub-file with '\input'
\begin{document}
% ...
\begin{figure}
  \input{subfile}
  \caption{A subfile}
\end{figure}
% ...
\end{document}
```

Example 2: Use of *standalone* class.

```
% A sub-file (e.g. picture) using the 'standalone' class:
% Use 'standalone' as document class:
\documentclass{standalone}

% Load packages needed for this TeX file:
\usepackage{tikz}

% Surround TeX code with 'document' environment as usually:
\begin{document}
% Add your TeX code, e.g. a picture:
\begin{tikzpicture}
  \draw (0,0) rectangle (2,1) node [midway] {Example};
\end{tikzpicture}
\end{document}
```

Example 3: Effective code if compiled standalone.

```
\documentclass{article}

\newenvironment{standalone}{\begin{preview}}{\end{preview}}
\input{standalone.cfg}
% which by defaults loads:
% \PassOptionsToPackage{active,tightpage}{preview}
\usepackage{preview}

\usepackage{tikz}

\begin{document}
\begin{standalone}
\begin{tikzpicture}
  \draw (0,0) rectangle (2,1) node [midway] {Example};
\end{tikzpicture}
\end{standalone}
\end{document}
```

Example 4: Effective code if included in a main document.

```
\begingroup
\begin{tikzpicture}
  \draw (0,0) rectangle (2,1) node [midway] {Example};
\end{tikzpicture}
\endgroup
\endinput
```

2.7 Support for Beamer Presentations

Presentation can be written in LaTeX using the `beamer` class. Each presentation frame is wrapped in a `frame` environment. Overlay effects can be added using special macros. This effects result in multiple pages per frame. Pictures with such overlay effects can not be compiled standalone using the normal settings. Instead the `standalone` class must load the `beamer` class and wrap the content also in a `frame` environment while skipping the `preview` environment. To activate this settings load the `standalone` class with the `beamer` option. Because the `frame` environment is quite special (it normally collects all it's content and calls the `\frame`) and must also support verbatim content it is not easily possible to redefined the `document` environment to include `frame`. Also `frame` accepts options which `document` doesn't. Therefore a second environment called `standaloneframe` is used in the beamer picture files. It will be equal to `frame` in standalone mode, but without effect otherwise.

`\ifstandalonebeamer`

This if switch is only true if the class is compiled with the `beamer` option. The package sets it to false. It can be used to place beamer specific options in the configuration files, which should be skipped for non-beamer standalone files.

Example 5: Use of `standalone` class with `beamer` option.

```
% Use of 'standalone' class with a beamer overlay:
\documentclass[beamer]{standalone}
% Load packages needed for this TeX file:
\usepackage{tikz}

% Surround TeX code with 'document' environment as usually:
\begin{document}
\begin{standaloneframe}[options] % e.g. 'fragile'
 % Add your TeX code:
   \only<1>{ One }%
   \only<2>{ Two }%
\end{standaloneframe}
\end{document}
```

Example 6: Effective beamer code if compiled standalone.

```
\documentclass{beamer}
\input{standalone.cfg}

\usepackage{tikz}

\begin{document}
\begin{frame}[your options]
  \only<1>{ One }%
  \only<2>{ Two }%
\end{frame}
\end{document}
```

Example 7: Effective code if included in a beamer presentation.

```
\begingroup
  \only<1>{ One }%
  \only<2>{ Two }%
\endgroup
\endinput
```

2.8 `standalone.tex`

Example 8: Usage of 'standalone.tex'.

```
\input{standalone} % use before any '\documentclass'
\ifstandalone
  % Used only if compiled standalone
\fi
```

2.9 Usage with `svn-multi` keywords

If the version control package `svn-multi` is used, the keyword macros (`\svnid` or `\svnidlong`) need to be placed after the `\begin{document}` to be taken into account by the main document. The `svn-multi` package must be loaded by the sub-file preamble to avoid compile errors in standalone mode. Alternative, if the keywords are not required in this mode, they can be surrounded by `\ifstandalone\else...\fi`.

3 Implementation

3.1 The Package File

The package file is to be loaded by a main document which includes `standalone` sub-files. It is also loaded by the `standalone` class to share code. The class then redefines certain macros.

3.1.1 If-Switches

`\ifstandalone` Declare `standalone` if-switch and set it to false. The class will set it to true. The `\csname` trickery is used to avoid issues if the switch was already defined.

```
\expandafter\newif\csname ifstandalone\endcsname
\standalonefalse
```

`\ifstandalonebeamer` Make sure that `standalonebeamer` if-switch is defined and set it to false. If the class was loaded beforehand with the `beamer` option it is already defined as true. The `\csname` trickery is used to avoid issues if the switch was already defined.

```
\@ifundefined{ifstandalonebeamer}{%
\expandafter\newif\csname ifstandalonebeamer\endcsname
\standalonebeamerfalse
}{}%
```

`\onlyifstandalone` Macro version of `\ifstandalone`. The `{ }` around the argument protects the content from the package etc. scanners. Only defined if not already defined by the class, in the case of a `standalone` file included other `standalone` files.

```
\@ifundefined{onlyifstandalone}
  {\let\onlyifstandalone\@gobble}
  {}
```

`\ifsa@subpreambles` The if-switches for the options.

`\ifsa@sortsubpreambles`

`\ifsa@printsubpreambles`

```
\newif\ifsa@subpreambles
\newif\ifsa@sortsubpreambles
\newif\ifsa@printsubpreambles
```

3.1.2 Options

```
\DeclareOption{subpreambles}{%
  \sa@subpreamblestrue
}
\DeclareOption{sort}{%
  \sa@subpreamblestrue
  \sa@sortsubpreamblestrue
}
\DeclareOption{print}{%
  \sa@subpreamblestrue
  \sa@printsubpreamblestrue
}
\DeclareOption{comments}{%
```

```
  \def\sa@percent{\@makeother\%}%
}
\DeclareOption{nocomments}{%
  \def\sa@percent{}%
}
\ProcessOptions*\relax
```

In non-sorted print mode comments are preserved by default.

```
\ifsa@printsubpreambles
  \ifsa@sortsubpreambles\else
    \@ifundefined{sa@percent}{%
      \def\sa@percent{\@makeother\%}%
    }{}%
  \fi
\fi
```

`\sa@filepath` File name macro. If the `fink` package is loaded the macros `\finkdir` (with leading '`./`' removed) and `\finkpath` is used, otherwise the LaTeX macro `\@filef@und` (with trailing space removed and with '.tex' added if it has no file extension). The latter causes issues if `\input` etc. was used before `\documentclass` in sub-files.

```
\@ifundefined{finkpath}{%
  \def\sa@rmspace#1 \empty{#1}%
  \def\sa@chkext#1.#2 \empty#3{%
    \ifx\empty#3\empty
      \sa@rmspace#1\empty.#2%
    \else
      #1.#2%
      \expandafter\sa@rmrest
    \fi
  }%
  \def\sa@rmrest tex \empty{}%
  \def\sa@filepath{\expandafter\sa@chkext\@filef@und\empty.tex \empty\empty}%
}{%
  \def\sa@filepath{\expandafter\expandafter\expandafter\sa@rmdotslash\expandafter\finkdir\fink
  \def\sa@rmdotslash#1./#2\empty{%
    \ifx\empty#1\empty
      \sa@@rmdotslash#2%
    \else
      \ifx\empty#2\empty
        #1%
      \else
        \sa@@rmdotslash#1./#2%
      \fi
    \fi
  }
  \def\sa@@rmdotslash#1./\empty{#1}%
}
```

3.1.3 Processing of Sub-Preambles

```
\ifsa@subpreambles
```

`\sa@out` Write handle.

```
\newwrite\sa@out
```

`\sa@write` Helper macro for file output.

```
\def\sa@write{\immediate\write\sa@out}%
\ifsa@printsubpreambles
```

`\sa@removeonlyifstandalone` Scans for `\onlyifstandalone` and removes it argument.

```
\long\def\sa@removeonlyifstandalone#1\onlyifstandalone{%
  \g@addto@macro\sa@preamble{#1}%
  \@ifnextchar\sa@endmarker
    {\@gobble}%
    {\expandafter\sa@gobbleeol\expandafter\sa@removeonlyifstandalone\expandafter^^J\@gobble}%
}
\fi
```

3.1.4 Sorting of package options

Macros only needed for this mode are defined inside the `\if...` to save memory otherwise.

```
\ifsa@sortsubpreambles
```

`\sa@usepackagewithoutoptions` Simply calls the original `\usepackage` while skipping the optional argument with potential package options.

```
\newcommand{\sa@usepackagewithoutoptions}[2][]{%
  \sa@orig@usepackage{#2}%
}
```

`\sa@endmarker` Unique end marker. Will not be expanded.

```
\def\sa@endmarker{%
  \@gobble{sa@endmarker}%
}
\ifsa@printsubpreambles
```

In sorted print mode all collected package etc. information is printed into the output file, followed by the reduced sub-preambles.

```
\AtEndDocument{%
  \sa@write{\@percentchar\space Packages required by sub-files:}%
  \expandafter\@for\expandafter\pkg\expandafter:\expandafter=\sa@collpkgs\do{%
    \ifx\pkg\empty\else
      \sa@write{%
        \string\usepackage%
        \expandafter\ifx\csname sa@pkgopts@\pkg\endcsname\empty\else%
          [\csname sa@pkgopts@\pkg\endcsname]%
```

```
        \fi
        {\pkg}%
        \expandafter\ifx\csname sa@pkgdate@\pkg\endcsname\relax\else%
          [\csname sa@pkgdate@\pkg\endcsname]%
        \fi
        }%
    \fi
  }%
  \ifx\sa@collpgflibs\empty\else
  \sa@write{^^J\@percentchar\space PGF libraries required by sub-files:}%
  \expandafter\@for\expandafter\lib\expandafter:\expandafter=\sa@collpgflibs\do{%
    \ifx\lib\empty\else
      \sa@write{\string\usepgflibrary{\lib}}%
    \fi
  }%
  \fi
  \ifx\sa@colltikzlibs\empty\else
  \sa@write{^^J\@percentchar\space TikZ libraries required by sub-files:}%
  \expandafter\@for\expandafter\lib\expandafter:\expandafter=\sa@colltikzlibs\do{%
    \ifx\lib\empty\else
      \sa@write{\string\usetikzlibrary{\lib}}%
    \fi
  }%
  \fi
  \ifx\sa@colltikztiminglibs\empty\else
  \sa@write{^^J\@percentchar\space TikZ-Timing libraries required by sub-files:}%
  \expandafter\@for\expandafter\lib\expandafter:\expandafter=\sa@colltikztiminglibs\do{%
    \ifx\lib\empty\else
      \sa@write{%
        \string\usetikztiminglibrary%
        \expandafter\ifx\csname sa@tikztimingopts@\lib\endcsname\empty\else%
          [\csname sa@tikztimingopts@\lib\endcsname]%
        \fi
        {\lib}%
        \expandafter\ifx\csname sa@tikztimingdate@\lib\endcsname\relax\else%
          [\csname sa@tikztimingdate@\lib\endcsname]%
        \fi
        }%
    \fi
  }%
  \fi
  \sa@write{\expandafter\unexpanded\expandafter{\sa@preamble}}%
  \message{^^JPackage 'standalone' INFO: See file '\jobname.stp' for list of sub-preambles.^^J
  \immediate\closeout\sa@out
}
```

`\sa@removepackages` Scans for `\usepackage`.

```
\long\def\sa@removepackages#1\usepackage{%
  \sa@removepgflibs#1\usepgflibrary\sa@endmarker
  \@ifnextchar\sa@endmarker
```

```
    {\@gobble}%
    {\sa@sortpackages}%
}
```

`\sa@removepgflibs` Scans for `\usepgflibrary`.

```
\long\def\sa@removepgflibs#1\usepgflibrary{%
  \sa@removetikzlibs#1\usetikzlibrary\sa@endmarker
  \@ifnextchar\sa@endmarker
    {\@gobble}%
    {\sa@sortpgflibs}%
}
```

`\sa@removetikzlibs` Scans for `\usetikzlibrary`.

```
\long\def\sa@removetikzlibs#1\usetikzlibrary{%
  \sa@removetikztiminglibs#1\usetikztiminglibrary\sa@endmarker
  \@ifnextchar\sa@endmarker
    {\@gobble}%
    {\sa@sorttikzlibs}%
}
```

`\sa@removetikztiminglibs` Scans for `\usetikztiminglibrary`.

```
\long\def\sa@removetikztiminglibs#1\usetikztiminglibrary{%
  \sa@removeonlyifstandalone#1\onlyifstandalone\sa@endmarker
  \@ifnextchar\sa@endmarker
    {\@gobble}%
    {\sa@sorttikztiminglibs}%
}
```

`\sa@sortpackage` Reads `\usepackage` arguments and stores them away. A list of all packages is compiled. Every package is only added once and has also a list of options used, also only saved once. If package dates are requested then the highest one is stored. Trailing newlines are removed.

```
\def\sa@collpkgs{}%
\newcommand\sa@sortpackages[2][]{%
  \@ifnextchar[%]
    {\sa@@sortpackages{#1}{#2}}%
    {\sa@@sortpackages{#1}{#2}[]}%
}
\def\sa@@sortpackages#1#2[#3]{%
  \@for\pkg:=#2\do {%
    \@ifundefined{sa@pkgopts@\pkg}%
      {%
        \expandafter\g@addto@macro\expandafter\sa@collpkgs\expandafter{\expandafter,\pkg}%
        \global\@namedef{sa@pkgopts@\pkg}{#1}%
        \global\@namedef{sa@pkgopt@\pkg @}{}%
        \ifx\relax#1\relax\else
          \@for\opt:=#1\do{\global\@namedef{sa@pkgopt@\pkg @\opt}{}}%
        \fi
      }%
```

```
      {%
        \ifx\relax#1\relax\else
          \@for\opt:=#1\do{%
            \@ifundefined{sa@pkgopt@\pkg @\opt}%
              {%
                \expandafter\g@addto@macro\csname sa@pkgopts@\pkg\expandafter\endcsname\expanda
                \global\@namedef{sa@pkgopt@\pkg @\opt}{}%
              }{}%
          }%
        \fi
      }%
    \ifx\relax#3\relax\else
    \@ifundefined{sa@pkgdate@\pkg}%
      {\global\@namedef{sa@pkgdate@\pkg}{#3}}%
      {%
        \begingroup
        \edef\@tempa{{\csname sa@pkgdate@\pkg\endcsname}{#3}}%
        \expandafter\sa@getlargerdate\@tempa
        \expandafter\xdef\csname sa@pkgdate@\pkg\endcsname{\sa@thedate}%
        \endgroup
      }%
    \fi
  }%
  \sa@gobbleeol\sa@removepackages^^J%
}
```

`\sa@getlargerdate` Takes two package dates and returns the larger one as `\sa@thedate`.

```
\def\sa@getlargerdate#1#2{%
  \sa@@getdate#1\relax\relax0/0/0\relax\empty\relax
  \let\sa@datea\sa@date
  \sa@@getdate#2\relax\relax0/0/0\relax\empty\relax
  \ifdim\sa@datea pt>\sa@date pt
    \def\sa@thedate{#1}%
  \else
    \def\sa@thedate{#2}%
  \fi
}
\def\sa@@getdate#1/#2/#3\relax{%
  \@ifnextchar\relax
    {%
      \def\sa@date{#1.#2#3}%
      \sa@@rmdate
    }%
    {%
      \def\sa@date{0}%
      \sa@@rmdate
    }%
}
\def\sa@@rmdate#1\empty\relax{}
```

`\sa@sortpgflibs` Reads the `\usepgflibrary` argument and stores it away. Trailing newlines are removed.

```
\def\sa@collpgflibs{}%
\def\sa@sortpgflibs#1{%
  \@for\lib:=#1\do {%
    \@ifundefined{sa@pgflib@\lib}%
      {%
        \expandafter\g@addto@macro\expandafter\sa@collpgflibs\expandafter{\expandafter,\lib}%
        \global\@namedef{sa@pgflib@\lib}{}%
      }%
      {}%
  }%
  \sa@gobbleeol\sa@removepgflibs^^J%
}
```

`\sa@sorttikzlibs` Reads the `\usetikzlibrary` argument and stores it away. Trailing newlines are removed.

```
\def\sa@colltikzlibs{}%
\def\sa@sorttikzlibs#1{%
  \@for\lib:=#1\do {%
    \@ifundefined{sa@tikzlib@\lib}%
      {%
        \expandafter\g@addto@macro\expandafter\sa@colltikzlibs\expandafter{\expandafter,\lib}%
        \global\@namedef{sa@tikzlib@\lib}{}%
      }%
      {}%
  }%
  \sa@gobbleeol\sa@removetikzlibs^^J%
}
```

`\sa@sorttikztiminglibs` Reads `\usetikztiminglibrary` arguments and stores them away. Trailing newlines are removed.

```
\def\sa@colltikztiminglibs{}%
\newcommand\sa@sorttikztiminglibs[2][]{%
  \@ifnextchar[%]
    {\sa@@sorttikztiminglibs{#1}{#2}}%
    {\sa@@sorttikztiminglibs{#1}{#2}[]}%
}
\def\sa@@sorttikztiminglibs#1#2[#3]{%
  \@for\lib:=#2\do {%
    \@ifundefined{sa@tikztimingopts@\lib}%
      {%
        \expandafter\g@addto@macro\expandafter\sa@colltikztiminglibs\expandafter{\expandafter,
        \global\@namedef{sa@tikztimingopts@\lib}{#1}%
        \global\@namedef{sa@tikztimingopt@\lib @}{}%
        \ifx\relax#1\relax\else
          \@for\opt:=#1\do{\global\@namedef{sa@tikztimingopt@\lib @\opt}{}}%
        \fi
      }%
```

```
      {%
        \ifx\relax#1\relax\else
          \@for\opt:=#1\do{%
            \@ifundefined{sa@tikztimingopt@\lib @\opt}%
              {%
                \expandafter\g@addto@macro\csname sa@tikztimingopts@\lib\expandafter\endcsname
                \global\@namedef{sa@tikztimingopt@\lib @\opt}{}%
              }{}%
          }%
        \fi
      }%
    \ifx\relax#3\relax\else
    \@ifundefined{sa@tikztimingdate@\lib}%
      {\global\@namedef{sa@tikztimingdate@\lib}{#3}}%
      {%
        \begingroup
        \edef\@tempa{{\csname sa@tikztimingdate@\lib\endcsname}{#3}}%
        \expandafter\sa@getlargerdate\@tempa
        \expandafter\xdef\csname sa@tikztimingdate@\lib\endcsname{\sa@thedate}%
        \endgroup
      }%
    \fi
  }%
  \sa@gobbleeol\sa@removetikztiminglibs^^J%
}
```

`\sa@gobbleopt` Gobbles an optional argument and a potential line endings and then executes the command given by `#1`.

```
\def\sa@gobbleopt#1[#2]{%
  \@ifnextchar^^J%
    {\sa@gobbleeol{#1}}{#1}%
}

\else
```

`\sa@scanpackages` Scans for `\usepackage`.

```
\def\sa@scanpackages#1\usepackage{%
  \@ifnextchar\sa@endmarker
    {\@gobble}%
    {\sa@collectpackage}
}
```

`\sa@collectpackage` Reads `\usepackage` arguments (ignores optional date) and stores it away. The options are later passed to the package to avoid option clashes.

```
\newcommand\sa@collectpackage[2][]{%
  \ifx\relax#1\relax\else
    \g@addto@macro\sa@collopts{\PassOptionsToPackage{#1}{#2}}%
  \fi
  \sa@scanpackages
```

```
}
\fi
```

`\sa@collopts` Accumulator for collected options. Is executed and cleared at the end of this package.

```
\def\sa@collopts{}
\AtEndOfPackage{\sa@collopts\let\sa@collopts\relax}
```

End of `\ifsa@sortsubpreambles`.

```
\fi
```

`standalonepreambles` This environment simply adds a group and sets the endline character to a printed newline and the argument character `#` as a normal character. The first suppresses `\par`'s in the stored sub-preambles while preserving newlines. The latter is required to permit macro arguments in the preambles. Otherwise a `#` is doubled to `##` causing compile errors when the sub-preambles are used. The `.sta` file is closed after this environment.

```
\def\standalonepreambles{%
  \begingroup
  \endlinechar=\m@ne
  \@makeother\#%
}
\def\endstandalonepreambles{%
  \endgroup
  \endinput
}
```

`subpreambles` This environment rereads the sub-preambles from the `.sta` files and stores it globally under the name "`\prevsubpreamble@`⟨*file name*⟩". If sorting is enabled the sub-preambles are also scanned for loaded packages.

```
\long\gdef\subpreamble#1#2\endsubpreamble{%
  \expandafter\gdef\csname prevsubpreamble@#1\endcsname{#2}%
  \ifsa@sortsubpreambles
    \sa@scanpackages#2\usepackage\sa@endmarker
  \fi
}
\def\endsubpreamble{}%
```

If in `print` mode open the `.stp` file.

```
\ifsa@printsubpreambles
  \immediate\openout\sa@out=\jobname.stp\relax
\else
```

otherwise:

Process `.sta` file from last run. All changes must be made by own macros which define the value globally. Therefore the input is wrapped in a group. Some spaces or special line endings could process typeset content, which causes errors inside the preamble. To be on the save side the input 'content' is stored in a temp box.

```
\begingroup
  \setbox\@tempboxa\hbox{%
  \InputIfFileExists{\jobname.sta}{}{\PackageInfo{standalone}{STA file not found!}{}{}}%
  }%
\endgroup
```

`\AtBeginDocument` At begin of the document the `.sta` file is read again. This time the sub-preamble macros are executed as normal. The `standalone` macros are defined to be without effect. If 'sorting' is enabled `\usepackage` is temporarily redefined to ignore any given options, which where already passed (`\PassOptionsToPackage`) beforehand.

```
\AtBeginDocument{%
  \let\subpreamble\@gobble
  \let\endsubpreamble\relax
  \let\standalonepreambles\relax
  \let\endstandalonepreambles\relax
  \ifsa@sortsubpreambles
    \let\sa@orig@usepackage\usepackage
    \let\usepackage\sa@usepackagewithoutoptions
  \fi
  \InputIfFileExists{\jobname.sta}{}{}%
  \ifsa@sortsubpreambles
    \let\usepackage\sa@orig@usepackage
  \fi
  \immediate\openout\sa@out=\jobname.sta\relax
  \immediate\write\sa@out{\string\standalonepreambles}%
}
```

`\AtEndDocument` At end of the document write end macro to and close `.sta` file.

```
\AtEndDocument{%
  \sa@write{\string\endstandalonepreambles}%
  \immediate\closeout\sa@out
}
```

End of `\ifsa@printsubpreambles`.

```
\fi
```

End of `\ifsa@subpreambles`.

```
\fi
```

3.1.5 Skipping of Sub-Preambles in Main Mode

This macros make the main document skip all preambles in sub-files.

`\sa@gobbleeol` Gobbles all following line endings (i.e. empty lines) and then executes the command given by `#1`. Because `\@ifnextchar` ignores spaces this also removes lines with only spaces.

```
\def\sa@gobbleeol#1^^J{%
  \@ifnextchar^^J%
    {\sa@gobbleeol{#1}}{#1}%
}
```

`\standaloneignore` This macro must only be used in a sub-file before a `\documentclass`. It gobbles everything up to this macro and then executes the `standalone` definition of it shown further below. It should be written as `\csname standaloneignore\endcsname` to ignore errors in standalone mode. The second definition allows the user to also write `\csname standaloneignore \endcsname` (note the extra space) without errors.

```
\long\def\standaloneignore#1\documentclass{%
  \sa@documentclass
}
\@namedef{standaloneignore\space}{\standaloneignore}
```

`\sa@documentclass` The `standalone` definition of `\documentclass`. If the sub-preambles are to be processed then the starting content is written into the output file etc., but only for the first time this sub-file is included. Some input related settings are set-up (line endings, macro argument and comments). Finally `\sa@gobble` is called to process the preamble.

```
\newcommand{\sa@documentclass}[2][]{%
  \let\document\sa@document
  \begingroup
  \ifsa@subpreambles
    \@ifundefined{sa@written@\sa@filepath}%
      {%
        \ifsa@printsubpreambles
          \ifsa@sortsubpreambles
            \begingroup
              \edef\@tempa{^^J\@percentchar\space Preamble from file '\sa@filepath'^^J}%
              \expandafter\g@addto@macro\expandafter\sa@preamble\expandafter{\@tempa}%
            \endgroup
          \else
            \sa@write{^^J\@percentchar\space Preamble from file '\sa@filepath'}%
          \fi
        \else
          \sa@write{\string\subpreamble{\sa@filepath}}%
        \fi
      }{}%
    \global\@namedef{subpreamble@\sa@filepath}{}%
    \ifsa@printsubpreambles
      \endlinechar=`\^^J%
    \else
      \endlinechar=\m@ne
    \fi
    \@makeother\#%
    \@nameuse{sa@percent}%
  \fi
  \def\sa@gobbleto{document}%
  \sa@gobbleeol\sa@gobble^^J%
}
```

`\sa@gobble` Gobbles everything to the next `\begin`, then checks if it was a `\begin{document}`.

If sub-preamble extraction is activated it accumulates the skipped content in macros named "`\subpreamble@`⟨*file name*⟩". Every sub-file is remembered and its preamble is only saved once. In `print` mode the file body is ignored and a appropriate warning is printed, otherwise the current and previous sub-preamble of the current processed file are compared. If different the file body is also ignored to avoid errors due to possible newly required but not loaded packages. The user is warned again about this and is asked to rerun LaTeX.

```
\def\sa@preamble{}%
\long\def\sa@gobble#1\begin#2{%
  \def\@tempa{#2}%
  \ifx\@tempa\sa@gobbleto
    \ifsa@subpreambles
      \expandafter\g@addto@macro\csname subpreamble@\sa@filepath\endcsname{#1}%
      \@ifundefined{sa@written@\sa@filepath}%
        {%
          \ifsa@printsubpreambles
            \ifsa@sortsubpreambles
              \sa@removepackages#1\usepackage\sa@endmarker
            \else
              \begingroup
              \let\sa@preamble\empty
              \sa@removeonlyifstandalone#1\onlyifstandalone\sa@endmarker
              \expandafter\sa@write\expandafter{\expandafter\unexpanded\expandafter{\sa@preambl
              \endgroup
            \fi
          \else
            \sa@write{\unexpanded{#1}}%
            \sa@write{\string\endsubpreamble}%
          \fi
        }{}%
      \global\@namedef{sa@written@\sa@filepath}{}%
      \ifsa@printsubpreambles
        \def\next{%
          \endgroup
          \PackageWarning{standalone}{Running 'standalone' package in sub-preamble print mode.
          \null
          \endinput
        }
      \else
      \expandafter
      \ifx
      \csname prevsubpreamble@\sa@filepath \expandafter\endcsname
      \csname     subpreamble@\sa@filepath \endcsname
        \def\next{\expandafter\endgroup\expandafter\begin\expandafter{\sa@gobbleto}}%
      \else
        %\expandafter\show\csname prevsubpreamble@\sa@filepath \endcsname
        %\expandafter\show\csname     subpreamble@\sa@filepath \endcsname
        \def\next{%
          \endgroup
```

```
          \PackageWarning{standalone}{Sub-preamble of file '\sa@filepath' has changed. Content
          \immediate\write\@mainaux{%
            \@percentchar\space standalone package info: Rerun LaTeX!
          }
          \null
          \endinput
        }
      \fi
      \fi
    \else
      \def\next{\expandafter\endgroup\expandafter\begin\expandafter{\sa@gobbleto}}%
    \fi
  \else
    \ifsa@subpreambles
      \expandafter\g@addto@macro\csname subpreamble@\sa@filepath\endcsname{#1\begin{#2}}%
      \@ifundefined{sa@written@\sa@filepath}%
        {\sa@write{\unexpanded{#1\begin{#2}}}}{}%
    \fi
    \def\next{\sa@gobble}%
  \fi
  \next
}
```

`standalone` Provide an empty definition of the `standalone` environment. The class is defining it with the code required in `standalone` mode.

```
\@ifundefined{standalone}
  {\newenvironment{standalone}[1][]{}{}}
  {}
```

`standalone` Provide an 'empty' definition of the `standaloneframe` environment. It only gobbles all arguments: `<...>[<...>][...]{...}{...}`. Please note that the last two `{ }` arguments are also optional. The class is defining it with the code required in `standalone` mode.

```
\@ifundefined{standaloneframe}
  {\@ifundefined{beamer@newenv}
    {\newenvironment{standaloneframe}[1][]{%
      \@ifnextchar[%]
        {\sa@framegobbleopt}{\sa@framegobbleargs}}{}%
    }
    {\newenvironment<>{standaloneframe}[1][]{%
      \@ifnextchar[%]
        {\sa@framegobbleopt}{\sa@framegobbleargs}}{}%
    }
   \def\sa@framegobbleopt[#1]{\sa@framegobbleargs}
   \def\sa@framegobbleargs{%
     \@ifnextchar\bgroup
       {\sa@framegobbleargs@}%
       {}%
   }
```

```
    \def\sa@framegobbleargs@#1{%
     \@ifnextchar\bgroup
       {\@gobble}%
       {}%
    }
   }
   {}
```

`\sa@orig@document` Store original `document` environment.

`\sa@orig@enddocument`

```
\let\sa@orig@document\document
\let\sa@orig@enddocument\enddocument
```

`\document` Redefine the `\begin{document}` of the main file to redefine `\documentclass`. This can not be done using `\AtBeginDocument` because the original redefines `\documentclass` itself after executing the hook.

```
\def\document{%
  \sa@orig@document
  \let\documentclass\sa@documentclass
  \ignorespaces
}
```

`\sa@document` This is the `\begin{document}` of the sub files. It does nothing except of redefining `\end{document}` and calling our own `atbegindocument` hook.

```
\def\sa@document{%
  \let\enddocument\sa@enddocument
  \sa@atbegindocument
}
```

`\sa@enddocument` This is the `\end{document}` of the sub files. It does nothing except of calling our own `atenddocument` hook and then the 'after end document' handler.

```
\def\sa@enddocument{%
  \sa@atenddocument
  \aftergroup\sa@@enddocument
}
```

`\sa@@enddocument` This is a 'after end document' handler for the sub-files. It restores macros and ends the input of the file.

```
\def\sa@@enddocument{%
  %\let\document\sa@orig@document
  \let\enddocument\sa@orig@enddocument
  \endinput
}
```

`\sa@atbegindocument` This hook simply ignores all spaces after `\begin{document}` in the sub files.

```
\def\sa@atbegindocument{%
  \ignorespaces
}%
```

`\sa@atenddocument` This hook simply ignores the last skip (normally the spaces) before `\end{document}` in the sub files.

```
\def\sa@atenddocument{%
  \unskip
}%
```

3.2 The Class File

3.2.1 If-Switches

`\ifstandalone` This if-switch is defined by both the class and package. This class sets it to true while the package (loaded by the main document) sets it to false.

```
\newif\ifstandalone
\standalonetrue
```

`\ifstandalonebeamer` This if-switch is defined by both the class and package. This class sets it to true only if the `beamer` option was given. The package (loaded by the main document) sets it always to false.

```
\newif\ifstandalonebeamer
\standalonebeamerfalse
```

`\onlyifstandalone` Macro version of `\ifstandalone`. The `{ }` around the argument protects the content from the package etc. scanners.

```
\let\onlyifstandalone\@firstofone
```

3.2.2 Options

```
\RequirePackage{kvoptions}
\SetupKeyvalOptions{prefix=sa@}
```

Use of `preview` package is optional but enabled by default. This defines the `\ifsa@preview` switch.

```
\DeclareBoolOption[true]{preview}
```

Enable beamer support.

```
\DeclareVoidOption{beamer}{%
  \def\sa@class{beamer}%
  \sa@previewfalse
  \standalonebeamertrue
}
```

Option to set underlying class. Default is `article`.

```
\DeclareStringOption[article]{class}
```

The rest of the options are accumulated and set as the official class options for the real class loaded afterwards. This avoids the passing of any `standalone` class options to the underlying class in any way. The `beamer` class for example has an option called '`class`' in a similar way the `standalone` class does, which would cause problems if not filtered out.

```
\def\sa@classoptionslist{}
\DeclareDefaultOption{%
```

```
  \xdef\sa@classoptionslist{\sa@classoptionslist,\CurrentOption}%
}
\ProcessKeyvalOptions*\relax
\let\@classoptionslist\sa@classoptionslist
```

Loads the class given by the `class` option with the rest of the options.

```
\begingroup
\xdef\@tempa{[\sa@classoptionslist]{\sa@class}}
\expandafter
\endgroup
\expandafter\LoadClass\@tempa
```

`standalone` The `standalone` environment is defined by default to be without effect. The `\endstandalone` macro is set to `\relax`, so a redefinition with `\renewenvironment` in the configuration file can be detected later.

```
\let\standalone\empty
\let\endstandalone\relax
```

Loads configuration file.

```
\input{standalone.cfg}
```

3.2.3 Preview Code

The `standalone` environment is redefined to use the `preview` environment as long it was not redefined in the configuration file.

```
\ifsa@preview
  \@ifundefined{endstandalone}{%
    \renewenvironment{standalone}
      {\preview   }
      {\endpreview}
  }{}%
  \RequirePackage{preview}
\fi
```

3.2.4 Beamer Frame Environment

```
\ifstandalonebeamer
```

`standaloneframe` Front-end for the beamer `frame` environment. Parses all arguments the same way and calls it with an added option.

```
\newenvironment{standaloneframe}{%
  \@ifnextchar<%
    {\@standaloneframe}%
    {\@@standaloneframe{}}%
}{\end{frame}}%
\def\@standaloneframe<#1>{%
    \@@standaloneframe{<#1>}%
}
\def\@@standaloneframe#1{%
  \@ifnextchar[%]
```

```
    {\@@@standaloneframe{#1}}%
    {\@@@standaloneframe{#1}[]}%
}%
\def\@@@standaloneframe#1[{%
  \@ifnextchar<%
    {\@@@@standaloneframe{#1}[]}%
    {\@@@@@@standaloneframe{#1}[]}%
}%
\def\@@@@standaloneframe#1[#2]{%
  \@ifnextchar[%]
    {\@@@@@standaloneframe{#1}{#2}}%
    {\begin{frame}#1[#2][environment=standaloneframe]}%
}%
\def\@@@@@standaloneframe#1#2[#3]{%
    \begin{frame}#1[#2][environment=standaloneframe,#3]%
}%
\def\@@@@@@standaloneframe#1[#2]{%
    \begin{frame}#1[environment=standaloneframe,#2]%
}%

\fi
```

3.2.5 Document Environment in Sub-Files

`\sa@cls@orig@document` Store original `document` environment.

`\sa@cls@orig@enddocument`

```
\let\sa@cls@orig@document\document
\let\sa@cls@orig@enddocument\enddocument
```

`document` Adds own 'after begin document' and 'before end document' hooks.

```
\def\document{%
  \sa@cls@orig@document
  \let\documentclass\sa@documentclass  % TODO: really required?
  \sa@cls@afterbegindocument
}
\def\enddocument{%
  \sa@cls@beforeenddocument
  \sa@cls@orig@enddocument
}
```

`\sa@cls@afterbegindocument` Hooks which add the `standalone` environment. Surrounding spaces are removed.

`\sa@cls@beforeenddocument` This hooks are used (instead of calling the content directly in the above macros) to add the possibility to fine tune this later, e.g. in the configuration file.

```
\def\sa@cls@afterbegindocument{\expandafter\standalone\ignorespaces}
\def\sa@cls@beforeenddocument{\unskip\endstandalone}
```

3.3 Simple TeX File

`\ifstandalone` Provides `\ifstandalone` switch which is `\iftrue` if the normal `\documentclass` was not yet executed (and subsequently redefined to be `\@twoclasseserror`).

```
\expandafter\ifx\csname ifstandalone\endcsname\relax
\expandafter\newif\csname ifstandalone\endcsname
\expandafter\ifx\csname @twoclasseserror\endcsname\documentclass
\else
  \standalonetrue
\fi
\fi
```

3.4 Config File

Default content of the configuration file. Users can override this by placing an own `standalone.cfg` file somewhere where TEX can find it (user `texmf` directory or local directory). This user file can load the default config file using using `\InputIfFileExists{standalone/standalone.cfg}{}{}`. Be default only the `preview` package option are set and the navigation symbols of beamer standalones are disabled.

```
\PassOptionsToPackage{active,tightpage}{preview}%

\ifstandalonebeamer
  \setbeamertemplate{navigation symbols}{}%
\fi
```

Change History

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.

U