

The *standalone* Class and Package

Martin Scharrer

martin@scharrer-online.de

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

Version v0.4 – 2011/02/28

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 lengthy, 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

class 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.

preview The boolean `preview` option can be used to disable the use of the `review` package. The default is `preview=true`. The package is not loaded if `preview=false` is set. The preview border can be set using the `border` option. This border will be added to the file content. The usage is either `border={<left> <lower> <right> <upper>}`, `border={<left/right> <lower/upper>}` or `border=<all sides>`. The default unit is bp (big points) which is the unit used by the PostScript and PDF formats. If `preview=false` is set this option will be ignored.

beamer 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 `standalone`). This file is then loaded or processed at the beginning of the next L^AT_EX 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

sort options inside the sub-preambles and/or the main preamble. 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 \LaTeX s `\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.

print 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 '`\langle main tex file name \rangle.stp`' (for *standalone*, *print*). 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 be 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 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.

print 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{\langle code \rangle}` (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 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 (*File Name Keeper*), 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

```
\begin{standalone}
  <sub-file content>
\end{standalone}
```

The `standalone` environment is automatically 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).

```
\begin{standaloneframe}<overlay specification>[<default overlay spec>]
  [<options>]{<optional frame title>}{<optional frame subtitle>}
  <code with beamer overlays>
\end{standaloneframe}
```

The `standaloneframe` environment must be used in sub-file holding `beamer` overlay code. It 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 but most might not be useful for normal sub-files. 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 simply 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.

```
1 % Main file
2 % Real document class:
3 \documentclass{article}
4
5 % Use the 'standalone' package:
6 \usepackage{standalone}
7
8 % Load all packages needed for all sub-files:
9 \usepackage{tikz}
10
11 % Inside the real 'document' environment
12 % read the sub-file with '\input'
13 \begin{document}
14 % ...
15 \begin{figure}
16   \input{subfile}
17   \caption{A subfile}
18 \end{figure}
19 % ...
20 \end{document}
```

Example 2: Use of *standalone* class.

```
21 % A sub-file (e.g. picture) using the 'standalone' class:
22 % Use 'standalone' as document class:
23 \documentclass{standalone}
24
25 % Load packages needed for this TeX file:
26 \usepackage{tikz}
27
28 % Surround TeX code with 'document' environment as usually:
29 \begin{document}
30 % Add your TeX code, e.g. a picture:
31 \begin{tikzpicture}
32   \draw (0,0) rectangle (2,1) node [midway] {Example};
33 \end{tikzpicture}
34 \end{document}
```

Example 3: Effective code if compiled standalone.

```
35 \documentclass{article}
36
37 \newenvironment{standalone}{\begin{preview}}{\end{preview}}
38 \input{standalone.cfg}
39 % which by defaults loads:
40 % \PassOptionsToPackage{active,tightpage}{preview}
41 \usepackage{preview}
42
43 \usepackage{tikz}
44
45 \begin{document}
46 \begin{standalone}
47 \begin{tikzpicture}
48   \draw (0,0) rectangle (2,1) node [midway] {Example};
49 \end{tikzpicture}
50 \end{standalone}
51 \end{document}
```

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

```
52 \begingroup
53 \begin{tikzpicture}
54   \draw (0,0) rectangle (2,1) node [midway] {Example};
55 \end{tikzpicture}
56 \endgroup
57 \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.

```
58 % Use of 'standalone' class with a beamer overlay:
59 \documentclass[beamer]{standalone}
60 % Load packages needed for this TeX file:
61 \usepackage{tikz}
62
63 % Surround TeX code with 'document' environment as usually:
64 \begin{document}
65 \begin{standaloneframe}[options] % e.g. 'fragile'
66   % Add your TeX code:
67   \only<1>{ One }%
68   \only<2>{ Two }%
69 \end{standaloneframe}
70 \end{document}
```

Example 6: Effective beamer code if compiled standalone.

```
71 \documentclass{beamer}
72 \input{standalone.cfg}
73
74 \usepackage{tikz}
75
76 \begin{document}
77 \begin{frame}[your options]
78   \only<1>{ One }%
79   \only<2>{ Two }%
80 \end{frame}
81 \end{document}
```

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

```
82 \begingroup
83   \only<1>{ One }%
84   \only<2>{ Two }%
85 \endgroup
86 \endinput
```

2.8 standalone.tex

Example 8: Usage of 'standalone.tex'.

```
87 \input{standalone} % use before any '\documentclass'  
88 \ifstandalone  
89 % Used only if compiled standalone  
90 \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.

```
91 \expandafter\newif\csname ifstandalone\endcsname
92 \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.

```
93 \@ifundefined{ifstandalonebeamer}{%
94 \expandafter\newif\csname ifstandalonebeamer\endcsname
95 \standalonebeamerfalse
96 }{}%
```

`\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.

```
97 \@ifundefined{onlyifstandalone}
98 {\let\onlyifstandalone\@gobble}
99 {}
```

`\ifsa@subpreambles`

```
\ifsa@sortsubpreambles
```

```
\ifsa@printsubpreambles
```

The if-switches for the options.

```
100 \newif\ifsa@subpreambles
101 \newif\ifsa@sortsubpreambles
102 \newif\ifsa@printsubpreambles
```

3.1.2 Options

```
103 \DeclareOption{subpreambles}{%
104   \sa@subpreambletrue
105 }
106 \DeclareOption{sort}{%
107   \sa@subpreambletrue
108   \sa@sortsubpreambletrue
109 }
110 \DeclareOption{print}{%
111   \sa@subpreambletrue
112   \sa@printsubpreambletrue
113 }
114 \DeclareOption{comments}{%
115   \def\sa@percent{\@makeother\}%
116 }
117 \DeclareOption{nocomments}{%
118   \def\sa@percent{}%
119 }
120 \DeclareOption{mode=none}{%
121   \let\sa@mode\relax
122 }
123 \DeclareOption{mode=pdf|tex}{%
124   \def\sa@mode{0}%
125 }
126 \DeclareOption{mode=tex}{%
127   \def\sa@mode{1}%
128 }
129 \DeclareOption{mode=pdf}{%
130   \def\sa@mode{2}%
131 }
132 \DeclareOption{mode=build}{%
```

```

133     \def\sa@mode{3}%
134 }
135 \DeclareOption{mode=buildnew}{%
136     \def\sa@mode{4}%
137 }
138 \ProcessOptions*\relax

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

139 \ifsa@printsubpreambles
140     \ifsa@sortsubpreambles\else
141         \@ifundefined{sa@percent}{%
142             \def\sa@percent{\@makeother\}%}%
143         }{}%
144     \fi
145 \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.

```

146 \@ifundefined{finkpath}{%
147     \def\sa@rmSPACE#1 \empty{#1}%
148     \def\sa@chkext#1.#2 \empty#3{%
149         \ifx\empty#3\empty
150             \sa@rmSPACE#1\empty.#2%
151         \else
152             #1.#2%
153             \expandafter\sa@rmrest
154         \fi
155     }%
156     \def\sa@rmrest tex \empty{}%
157     \def\sa@filepath{\expandafter\sa@chkext\@filef@und\empty.\%
158 }{%
159     \def\sa@filepath{\expandafter\expandafter\expandafter\%
160         sa@rmdotslash\expandafter\finkdir\finkfile\empty.\%
161         empty}%
162     \def\sa@rmdotslash#1./#2\empty{%
163         \ifx\empty#1\empty
164             \sa@@rmdotslash#2%
165         \else

```

```

164     \ifx\empty#2\empty
165     #1%
166     \else
167     \sa@@rmdotslash#1./#2%
168     \fi
169     \fi
170 }
171 \def\sa@@rmdotslash#1./\empty{#1}%
172 }

```

3.1.3 Processing of Sub-Preambles

```

173 \ifsa@subpreambles

```

\sa@out

Write handle.

```

174 \newwrite\sa@out

```

\sa@write

Helper macro for file output.

```

175 \def\sa@write{\immediate\write\sa@out}%

```

```

176 \ifsa@printssubpreambles

```

\sa@removeonlyifstandalone

Scans for \onlyifstandalone and removes it argument.

```

177 \long\def\sa@removeonlyifstandalone#1\onlyifstandalone{%
178   \g@addto@macro\sa@preamble{#1}%
179   \@ifnextchar\sa@endmarker
180     {\@gobble}%
181     {\expandafter\sa@gobbleeol\expandafter\
182       sa@removeonlyifstandalone\expandafter^^J\@gobble}%
183 }

```

```

183 \fi

```

3.1.4 Sorting of package options

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

```
184 \ifsa@sortsubpreambles
```

`\sa@usepackagewithoutoptions`

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

```
185 \newcommand{\sa@usepackagewithoutoptions}[2][\]{%
186   \sa@orig@usepackage{#2}%
187 }
```

`\sa@endmarker`

Unique end marker. Will not be expanded.

```
188 \def\sa@endmarker{%
189   \@gobble{sa@endmarker}%
190 }
```

```
191 \ifsa@printsubpreambles
```

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

```
192 \AtEndDocument{%
193   \sa@write{\@percentchar\space Packages required by sub-
194     files:}%
195   \expandafter\@for\expandafter\pkg\expandafter:\
196     expandafter=\sa@collpkgs\do{%
197     \ifx\pkg\empty\else
198       \sa@write{%
199         \string\usepackage%
200         \expandafter\ifx\csname sa@pkgopts@\pkg\endcsname\
201         empty\else%
202         [\csname sa@pkgopts@\pkg\endcsname]%
203         \fi
204         {\pkg}%
205         \expandafter\ifx\csname sa@pkgdate@\pkg\endcsname\
206         relax\else%
207         [\csname sa@pkgdate@\pkg\endcsname]%
208         \fi
209       }%
210     }
```

```

206     \fi
207 }%
208 \ifx\sa@collpgflibs\empty\else
209 \sa@write{^^J\@percentchar\space PGF libraries required ✓
    by sub-files:}%
210 \expandafter\@for\expandafter\lib\expandafter:\✓
    expandafter=\sa@collpgflibs\do{%
211     \ifx\lib\empty\else
212     \sa@write{\string\usepgflibrary{\lib}}%
213     \fi
214 }%
215 \fi
216 \ifx\sa@colltikzlibs\empty\else
217 \sa@write{^^J\@percentchar\space TikZ libraries required ✓
    by sub-files:}%
218 \expandafter\@for\expandafter\lib\expandafter:\✓
    expandafter=\sa@colltikzlibs\do{%
219     \ifx\lib\empty\else
220     \sa@write{\string\usetikzlibrary{\lib}}%
221     \fi
222 }%
223 \fi
224 \ifx\sa@colltikztiminglibs\empty\else
225 \sa@write{^^J\@percentchar\space TikZ-Timing libraries ✓
    required by sub-files:}%
226 \expandafter\@for\expandafter\lib\expandafter:\✓
    expandafter=\sa@colltikztiminglibs\do{%
227     \ifx\lib\empty\else
228     \sa@write{%
229         \string\usetikztiminglibrary%
230         \expandafter\ifx\csname sa@tikztimingopts@\lib\✓
            endcsname\empty\else%
231             [\csname sa@tikztimingopts@\lib\endcsname]%
232         \fi
233         {\lib}%
234         \expandafter\ifx\csname sa@tikztimingdate@\lib\✓
            endcsname\relax\else%
235             [\csname sa@tikztimingdate@\lib\endcsname]%
236         \fi
237     }%
238 \fi
239 }%
240 \fi
241 \sa@write{\expandafter\unexpanded\expandafter{\✓

```

```

    sa@preamble}}%
242 \message{^^JPackage 'standalone' INFO: See file '\jobname\
    .stp' for list of sub-preambles.^^J}%
243 \immediate\closeout\sa@out
244 }

```

\sa@removepackages

Scans for \usepackage.

```

245 \long\def\sa@removepackages#1\usepackage{%
246 \sa@removepgflibs#1\usepgflibrary\sa@endmarker
247 \@ifnextchar\sa@endmarker
248 {\@gobble}%
249 {\sa@sortpackages}%
250 }

```

\sa@removepgflibs

Scans for \usepgflibrary.

```

251 \long\def\sa@removepgflibs#1\usepgflibrary{%
252 \sa@removetikzlibs#1\usetikzlibrary\sa@endmarker
253 \@ifnextchar\sa@endmarker
254 {\@gobble}%
255 {\sa@sortpgflibs}%
256 }

```

\sa@removetikzlibs

Scans for \usetikzlibrary.

```

257 \long\def\sa@removetikzlibs#1\usetikzlibrary{%
258 \sa@removetikztiminglibs#1\usetikztiminglibrary\
    sa@endmarker
259 \@ifnextchar\sa@endmarker
260 {\@gobble}%
261 {\sa@sorttikzlibs}%
262 }

```


`\sa@removetikztiminglibs`

Scans for `\usetikztiminglibrary`.

```
263 \long\def\s@removetikztiminglibs#1\usetikztiminglibrary{%
264   \sa@removeonlyifstandalone#1\onlyifstandalone\✓
      sa@endmarker
265   \@ifnextchar\s@endmarker
266     {\@gobble}%
267     {\sa@sorttikztiminglibs}%
268 }
```

`\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.

```
269 \def\s@collpkgs{}%
270 \newcommand\s@sortpackages [2] []{%
271   \@ifnextchar [%]
272     {\sa@@sortpackages {#1}{#2}}%
273     {\sa@@sortpackages {#1}{#2} []}%
274 }
275 \def\s@@@sortpackages#1#2[#3]{%
276   \@for\pkg:=#2\do {%
277     \@ifundefined{sa@pkgopts@\pkg}%
278       {%
279         \expandafter\g@addto@macro\expandafter\s@collpkgs\✓
           expandafter{\expandafter , \pkg}%
280         \global\@namedef{sa@pkgopts@\pkg}{#1}%
281         \global\@namedef{sa@pkgopt@\pkg @}{}%
282         \ifx\relax#1\relax\else
283           \@for\opt:=#1\do{\global\@namedef{sa@pkgopt@\pkg ✓
             @\opt}{}}%
284         \fi
285       }%
286     {%
287       \ifx\relax#1\relax\else
288         \@for\opt:=#1\do{%
289           \@ifundefined{sa@pkgopt@\pkg @\opt}%
290             {%
291               \expandafter\g@addto@macro\csname ✓
                 sa@pkgopts@\pkg\expandafter\endcsname\✓
                 expandafter{\expandafter , \opt}%
```

```

292         \global\@namedef{sa@pkgopt@\pkg @\opt}{}%
293     }{}%
294 }%
295 \fi
296 }%
297 \ifx\relax#3\relax\else
298 \@ifundefined{sa@pkgdate@\pkg}%
299     {\global\@namedef{sa@pkgdate@\pkg}{#3}}%
300     {%
301         \ifnum\expandafter\expandafter
302             \expandafter\sa@@getdate\csname sa@pkgdate@\pkg\
303                 endcsname//00\relax<\sa@@getdate#3//00\relax
304             \global\@namedef{sa@pkgdate@\pkg}{#3}%
305         \fi
306     }%
307 \fi
308 \sa@gobbleeol\sa@removepackages^^J%
309 }
310 \def\sa@@getdate#1/#2/#3#4#5\relax{#1#2#3#4}

```

`\sa@sortpgflibs`

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

```

311 \def\sa@collpgflibs{}%
312 \def\sa@sortpgflibs#1{%
313     \@for\lib:=#1\do {%
314         \@ifundefined{sa@pgflib@\lib}%
315         {%
316             \expandafter\g@addto@macro\expandafter\
317                 sa@collpgflibs\expandafter{\expandafter ,\lib}%
318             \global\@namedef{sa@pgflib@\lib}{}%
319         }%
320     }%
321 \sa@gobbleeol\sa@removepgflibs^^J%
322 }

```

`\sa@sorttikzlibs`

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

```

323 \def\sacolltikzlibs{}%
324 \def\sasorttikzlibs#1{%
325   \@for\lib:=#1\do {%
326     \@ifundefined{sa@tikzlib@\lib}%
327     {%
328       \expandafter\g@addto@macro\expandafter\
329       sa@colltikzlibs\expandafter{\expandafter,\lib}%
330       \global\@namedef{sa@tikzlib@\lib}{}%
331     }%
332   }%
333   \sa@gobbleeol\saremovetikzlibs^^J%
334 }

```

\sa@sorttikztiminglibs

Reads \usetikztiminglibrary arguments and stores them away. Trailing newlines are removed.

```

335 \def\sacolltikztiminglibs{}%
336 \newcommand\sasorttikztiminglibs[2][{}]{%
337   \@ifnextchar[%]
338   {\sa@@sorttikztiminglibs{#1}{#2}}%
339   {\sa@@sorttikztiminglibs{#1}{#2}[]}%
340 }
341 \def\sasorttikztiminglibs#1#2[#3]{%
342   \@for\lib:=#2\do {%
343     \@ifundefined{sa@tikztimingopts@\lib}%
344     {%
345       \expandafter\g@addto@macro\expandafter\
346       sa@colltikztiminglibs\expandafter{\expandafter,\
347       lib}%
348       \global\@namedef{sa@tikztimingopts@\lib}{#1}%
349       \global\@namedef{sa@tikztimingopt@\lib @}{}%
350       \ifx\relax#1\relax\else
351         \@for\opt:=#1\do{\global\@namedef{\
352         sa@tikztimingopt@\lib @\opt}{}}%
353       \fi
354     }%
355     {%
356       \ifx\relax#1\relax\else
357         \@for\opt:=#1\do{%
358           \@ifundefined{sa@tikztimingopt@\lib @\opt}%
359           {%

```

```

357     \expandafter\g@addto@macro\csname ✓
        sa@tikztimingopts@\lib\expandafter\✓
        endcsname\expandafter{\expandafter ,\opt}✓
        %
358     \global\@namedef{sa@tikztimingopt@\lib @\✓
        opt}{}%
359     }{}%
360     }%
361     \fi
362     }%
363     \ifx\relax#3\relax\else
364     \@ifundefined{sa@tikztimingdate@\lib}%
365     {\global\@namedef{sa@tikztimingdate@\lib}{#3}}%
366     {%
367     \begingroup
368     \edef\@tempa{{\csname sa@tikztimingdate@\lib\✓
        endcsname}{#3}}%
369     \expandafter\sa@getlargerdate\@tempa
370     \expandafter\xdef\csname sa@tikztimingdate@\lib\✓
        endcsname{\sa@thedata}%
371     \endgroup
372     }%
373     \fi
374     }%
375     \sa@gobbleeol\sa@removetikztiminglibs^^J%
376 }

```

`\sa@gobbleopt`

Gobbles an optional argument and a potential line endings and then executes the command given by #1.

```

377 \def\s@gobbleopt#1[#2]{%
378   \@ifnextchar^^J%
379   {\sa@gobbleeol{#1}}{#1}%
380 }
381 \else

```

`\sa@scanpackages`

Scans for `\usepackage`.

```

382 \def\s@scanpackages#1\usepackage{%
383   \@ifnextchar\s@endmarker
384     {\@gobble}%
385     {\s@collectpackage}
386 }

```

\s@collectpackage

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

```

387 \newcommand\s@collectpackage [2] []{%
388   \ifx\relax#1\relax\else
389     \g@addto@macro\s@collopts{\PassOptionsToPackage
390       {#1}{#2}}%
391   \fi
392 }
393 \fi

```

\s@collopts

Accumulator for collected options. Is executed and cleared at the end of this package.

```

394 \def\s@collopts{}
395 \AtEndOfPackage{\s@collopts\let\s@collopts\relax}
396
397 End of \ifsa@sortsubpreambles.
398
399 \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.

```

397 \def\standalonepreambles{%
398   \begingroup
399   \endlinechar=\m@ne
400   \@makeother\#%
401 }
402 \def\endstandalonepreambles{%

```

```

403 \endgroup
404 \endinput
405 }

```

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.

```

406 \long\gdef\subpreamble#1#2\endsubpreamble{%
407 \expandafter\gdef\csname prevsubpreamble@#1\endcsname{#2}✓
    %
408 \ifsa@sortsubpreambles
409 \sa@scanpackages#2\usepackage\sa@endmarker
410 \fi
411 }
412 \def\endsubpreamble{ }%

```

If in print mode open the .stp file.

```

413 \ifsa@printsubpreambles
414 \immediate\openout\sa@out=\jobname.stp\relax
415 \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 safe side the input ‘content’ is stored in a temp box.

```

416 \begingroup
417 \setbox\@tempboxa\hbox{%
418 \InputIfFileExists{\jobname.sta}{\PackageInfo{✓
    standalone}{STA file not found!}}{}}%
419 }%
420 \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 were already passed (\PassOptionsToPackage) beforehand.

```

421 \AtBeginDocument{%
422   \let\subpreamble\@gobble
423   \let\endsubpreamble\relax
424   \let\standalonepreambles\relax
425   \let\endstandalonepreambles\relax
426   \ifsa@sortsubpreambles
427     \let\sa@orig@usepackage\usepackage
428     \let\usepackage\sa@usepackagewithoutoptions
429   \fi
430   \InputIfFileExists{\jobname.sta}{\relax}{\relax}%
431   \ifsa@sortsubpreambles
432     \let\usepackage\sa@orig@usepackage
433   \fi
434   \immediate\openout\sa@out=\jobname.sta\relax
435   \immediate\write\sa@out{\string\standalonepreambles}%
436 }

```

\AtEndDocument

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

```

437 \AtEndDocument{%
438   \sa@write{\string\endstandalonepreambles}%
439   \immediate\closeout\sa@out
440 }

```

End of \ifsa@printsubpreambles.

```

441 \fi

```

End of \ifsa@subpreambles.

```

442 \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.

```

443 \def\sa@gobbleeol#1^^J{%
444   \@ifnextchar^^J%
445     {\sa@gobbleeol{#1}}{#1}%
446 }

```

`\sa@gobbleline`

Gobbles the rest of the input line. This is required when skipping the body of a file to also skip everything on the same line after `\begin{document}`.

```
447 \def\s@gobbleline#1^^J{}%
```

`\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.

```
448 \long\def\standaloneignore#1\documentclass{%
449   \sa@documentclass
450 }
451 \@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.

```
452 \newcommand{\sa@documentclass}[2][ ]{%
453   \let\document\sa@document
454   \begingroup
455   \ifsa@subpreambles
456     \@ifundefined{sa@written@\sa@filepath}{%
457       {%
458         \ifsa@printssubpreambles
459           \ifsa@sortsubpreambles
460             \begingroup
461               \edef\@tempa{^^J\@percentchar\space Preamble ✓
462                 from file '\sa@filepath'^^J}%
463               \expandafter\g@addto@macro\expandafter\
464                 sa@preamble\expandafter{\@tempa}%
465             \endgroup
466           \else
467             \sa@write{^^J\@percentchar\space Preamble from ✓
468               file '\sa@filepath'}%
```



```

466         \fi
467     \else
468         \sa@write{\string\subpreamble{\sa@filepath}}%
469     \fi
470 }{}%
471 \global\@namedef{subpreamble@\sa@filepath}{}%
472 \ifsa@printsubpreambles
473     \endlinechar='^^J%
474 \else
475     \endlinechar=\m@ne
476 \fi
477 \@makeother\#%
478 \@nameuse{sa@percent}%
479 \fi
480 \def\sa@gobbleto{document}%
481 \sa@gobbleeol\sa@gobble^^J%
482 }

```

\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`.

```

483 \def\sa@preamble{}%
484 \long\def\sa@gobble#1\begin#2{%
485     \def\@tempa{#2}%
486     \ifx\@tempa\sa@gobbleto
487         \ifsa@subpreambles
488             \expandafter\g@addto@macro\csname subpreamble@\↵
489                 sa@filepath\endcsname{#1}%
490             \@ifundefined{sa@written@\sa@filepath}%
491                 {%
492                     \ifsa@printsubpreambles
493                     \ifsa@sortsubpreambles
494                     \sa@removepackages#1\usepackage\sa@endmarker
495                     \else
496                     \begingroup
497                     \let\sa@preamble\empty

```

```

497         \sa@removeonlyifstandalone#1\onlyifstandalone↵
           \sa@endmarker
498         \expandafter\sa@write\expandafter{\↵
           expandafter\unexpanded\expandafter{\↵
           sa@preamble}}}%
499         \endgroup
500         \fi
501     \else
502         \sa@write{\unexpanded{#1}}%
503         \sa@write{\string\endsubpreamble}%
504     \fi
505 }{}%
506 \global\@namedef{sa@written@\sa@filepath}{}%
507 \ifsa@printssubpreambles
508     \def\next{%
509         \endgroup
510         \PackageWarning{standalone}{Running 'standalone' ↵
           package in sub-preamble print mode. All body ↵
           content of file '\sa@filepath' is ignored↵
           !}{}{}%
511         \hbox to 1pt{\vbox to 1pt{}}%
512         \endinput
513         %\sa@gobbleline
514     }%
515 \else
516 \expandafter
517 \ifx
518 \csname prevsubpreamble@\sa@filepath \expandafter\↵
           endcsname
519 \csname      subpreamble@\sa@filepath \endcsname
520     \def\next{\expandafter\endgroup\expandafter\begin\↵
           expandafter{\sa@gobbleto}}}%
521 \else
522     %\expandafter\show\csname prevsubpreamble@\↵
           sa@filepath \endcsname
523     %\expandafter\show\csname      subpreamble@\↵
           sa@filepath \endcsname
524     \def\next{%
525         \endgroup
526         \PackageWarning{standalone}{Sub-preamble of file ↵
           '\sa@filepath' has changed. Content will be ↵
           ignored. Please rerun LaTeX!}{}{}%
527     \immediate\write\@mainaux{%
528         \@percentchar\space standalone package info: ↵

```

```

                    Rerun LaTeX!
529             }%
530             \hbox to 1pt{\vbox to 1pt{}}%
531             \endinput
532             %\sa@gobbleline
533         }%
534     \fi
535     \fi
536 \else
537     \def\next{\expandafter\endgroup\expandafter\begin\↵
        expandafter{\sa@gobbleto}}%
538 \fi
539 \else
540     \ifsa@subpreambles
541         \expandafter\g@addto@macro\csname subpreamble@\↵
            sa@filepath\endcsname{#1\begin{#2}}%
542         \@ifundefined{sa@written@\sa@filepath}%
543             {\sa@write{\unexpanded{#1\begin{#2}}}}{}%
544         \fi
545         \def\next{\sa@gobble}%
546     \fi
547 \next
548 }

```

standalone

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

```

549 \@ifundefined{standalone}
550     {\newenvironment{standalone}[1][[]]{}{}}
551     {}

```

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.

```

552 \@ifundefined{standaloneframe}
553     {\@ifundefined{beamer@newenv}
554         {\newenvironment{standaloneframe}[1][[]]{%
555             \@ifnextchar[%]
556                 {\sa@framegobbleopt}{\sa@framegobbleargs}}{}}%
557     }

```

```

558     {\newenvironment <>{standaloneframe}[1][ ]{%
559         \@ifnextchar [%]
560             {\sa@framegobbleopt}{\sa@framegobbleargs}}{}}%
561     }
562     \def \sa@framegobbleopt [#1]{\sa@framegobbleargs}
563     \def \sa@framegobbleargs{%
564         \@ifnextchar \bgroup
565             {\sa@framegobbleargs@}%
566             {}%
567     }
568     \def \sa@framegobbleargs@#1{%
569         \@ifnextchar \bgroup
570             {\@gobble}%
571             {}%
572     }
573     }
574     {}

```

`\sa@orig@document`

`\sa@orig@enddocument`

Store original document environment.

```

575 \let \sa@orig@document \document
576 \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.

```

577 \def \document{%
578     \sa@orig@document
579     \let \documentclass \sa@documentclass
580     \ignorespaces
581 }

```

`\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.

```
582 \def\s@document{%  
583   \let\enddocument\s@enddocument  
584   \sa@atbegindocument  
585 }
```

`\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.

```
586 \def\s@enddocument{%  
587   \sa@atenddocument  
588   \aftergroup\s@@enddocument  
589 }
```

`\sa@@enddocument`

This is a ‘after end document’ handler for the sub-files. It restores macros and ends the input of the file.

```
590 \def\s@@enddocument{%  
591   %\let\document\s@orig@document  
592   \let\enddocument\s@orig@enddocument  
593   \endinput  
594 }
```

`\sa@atbegindocument`

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

```
595 \def\s@atbegindocument{%  
596   \ignorespaces  
597 }%
```

`\sa@atenddocument`

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

```

598 \def\sa@atenddocument{%
599     \unskip
600 }%

```

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.

```

601 \newif\ifstandalone
602 \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.

```

603 \newif\ifstandalonebeamer
604 \standalonebeamerfalse

```

`\onlyifstandalone`

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

```

605 \let\onlyifstandalone\@firstofone

```

3.2.2 Options

```

606 \RequirePackage{kvoptions}
607 \SetupKeyvalOptions{prefix=sa@}

```

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

```

608 \DeclareBoolOption[true]{preview}

```

To set the preview border:

```

609 \DeclareStringOption{border}
610 \let\sa@border\relax

```

Enable beamer support.

```
611 \DeclareVoidOption{beamer}{%
612   \def\sa@class{beamer}%
613   \sa@previewfalse
614   \standalonebeamertrue
615 }
```

Option to set underlying class. Default is article.

```
616 \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.

```
617 \def\sa@classoptionslist{}
618 \DeclareDefaultOption{%
619   \xdef\sa@classoptionslist{\sa@classoptionslist,\↵
        CurrentOption}%
620 }
621 \ProcessKeyvalOptions*\relax
622 \let\@classoptionslist\sa@classoptionslist
```

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

```
623 \begingroup
624 \xdef\@tempa{[\sa@classoptionslist]{\sa@class}}
625 \expandafter
626 \endgroup
627 \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.

```
628 \let\standalone\empty
629 \let\endstandalone\relax
```

Loads configuration file.

```
630 \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.

```
631 \ifsa@preview
632   \@ifundefined{endstandalone}{%
633     \renewenvironment{standalone}
634       {\preview   }
635       {\endpreview}
636   }{}%
637   \RequirePackage{preview}
```

Read the preview border values. The default unit is bp (PostScript points). Allowed are one, two or four values.

```
638   \ifx\sa@border\relax\else
639     \begingroup
640     \def\rem@bp#1bp\relax#2\@nnil{#1}%
641     \def\default@bp#1{\expandafter\rem@bp\the\dimexpr#1 bp\relax bp\relax\@nnil}%
642     \def\sa@readborder#1 #2 #3 #4 {%
643       \ifx\#2#3#4\%
644         \@defaultunits\global\PreviewBorder=\dimexpr#1 bp\relax\@nnil%
645       \else
646         \ifx\#4\%
647           \xdef\PreviewBbAdjust{-\default@bp{#1} -\default@bp{#2} \default@bp{#1} \default@bp{#2}}%
648         \else
649           \xdef\PreviewBbAdjust{-\default@bp{#1} -\default@bp{#2} \default@bp{#3} \default@bp{#4}}%
650         \fi\fi
651       }%
652       \@firstofone{\expandafter\sa@readborder\sa@border} {} \relax
653     \endgroup
654   \fi
655 \fi
```

3.2.4 Beamer Frame Environment

```
656 \ifstandalonebeamer
```

`standaloneframe`

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

```

657 \newenvironment{standaloneframe}{%
658   \@ifnextchar<%
659     {\@standaloneframe}%
660     {\@@standaloneframe{}}}%
661 }{\end{frame}}%
662 \def\@standaloneframe<#1>{%
663   \@@standaloneframe{<#1>}%
664 }
665 \def\@@standaloneframe#1{%
666   \@ifnextchar[%]
667     {\@@@standaloneframe{#1}}%
668     {\@@@standaloneframe{#1}[]}%
669 }%
670 \def\@@@standaloneframe#1[{%
671   \@ifnextchar<%
672     {\@@@standaloneframe{#1}[]}%
673     {\@@@@standaloneframe{#1}[]}%
674 }%
675 \def\@@@@standaloneframe#1[#2]{%
676   \@ifnextchar[%]
677     {\@@@@standaloneframe{#1}{#2}}%
678     {\begin{frame}#1[#2][environment=standaloneframe]}%
679 }%
680 \def\@@@@@standaloneframe#1#2[#3]{%
681   \begin{frame}#1[#2][environment=standaloneframe ,#3]%
682 }%
683 \def\@@@@@@standaloneframe#1[#2]{%
684   \begin{frame}#1[environment=standaloneframe ,#2]%
685 }%
686 \fi

```

3.2.5 Document Environment in Sub-Files

`\sa@cls@orig@document`

`\sa@cls@orig@enddocument`

Store original document environment.

```

687 \let\sa@cls@orig@document\document
688 \let\sa@cls@orig@enddocument\enddocument

```

document

Adds own ‘after begin document’ and ‘before end document’ hooks.

```

689 \def\document{%
690   \sa@cls@orig@document
691   \let\documentclass\sa@documentclass % TODO: really ✓
        required?
692   \sa@cls@afterbegindocument
693 }
694 \def\enddocument{%
695   \sa@cls@beforeenddocument
696   \sa@cls@orig@enddocument
697 }

```

\sa@cls@afterbegindocument

\sa@cls@beforeenddocument

Hooks which add the standalone environment. Surrounding spaces are removed. 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.

```

698 \def\sa@cls@afterbegindocument{\expandafter\standalone\✓
        ignorespaces}
699 \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).

```

700 \expandafter\ifx\csname ifstandalone\endcsname\relax
701 \expandafter\newif\csname ifstandalone\endcsname
702 \expandafter\ifx\csname @twoclasseserror\endcsname\✓
        documentclass
703 \else

```

```

704 \standalonetrue
705 \fi
706 \fi

707 \expandafter\ifx\csname sa@mode\endcsname\relax\else

```

<code>\includestandalone</code>

```

708 \RequirePackage{ginc1tex}
709 \RequirePackage{ifpdf}
710 \ifpdf
711     \def\sa@graphicext{.pdf}
712 \else
713     \def\sa@graphicext{.eps}
714 \fi

715
716 \def\sa@comparegranpic#1{%
717     \begingroup
718     \let\next\empty
719     \expandafter\sa@comparegranpic@\pdffilemoddate{#1}\
720     sa@graphicext}\relax
721     \let\sa@picdate\sa@date
722     \let\sa@pictime\sa@time
723     \expandafter\sa@comparegranpic@\pdffilemoddate{#1.tex}\
724     relax
725     \ifnum\sa@date>\sa@picdate\relax
726         \def\next{\sa@buildgraphic{#1}}%
727     \else
728         \ifnum\sa@date=\sa@picdate\relax
729             \ifnum\sa@time>\sa@pictime\relax
730                 \def\next{\sa@buildgraphic{#1}}%
731             \fi
732         \fi
733     \fi
734     \expandafter
735     \endgroup
736     \next
737 }

738 \begingroup
739 \@makeother\D
740 \@makeother:

```

```

740 \global\@namedef{sa@comparegranpic@}D:#1#2#3#4#5#6#7#8#9\relax{
741     \def\sa@date{#1#2#3#4#5#6#7#8}%
742     \sa@comparegranpic@@#9\relax
743 }
744 \endgroup
745
746 \def\sa@comparegranpic@@#1#2#3#4#5#6#7\relax{%
747     \def\sa@time{#1#2#3#4#5#6}%
748     \def\sa@tz{#7}%
749 }
750
751 \def\sa@compilecmd{\ifpdf pdf\fi latex --interaction=
752     batchmode \image}%
753
754 \def\sa@buildgraphic#1{%
755     \ifeof18
756         \PackageError{standalone}{Shell escape needed to
757             create graphic!}{}%
758     \else
759         \begingroup
760         \def\image{#1}%
761         \immediate\write18{\sa@compilecmd}%
762         \endgroup
763     \fi
764 }
765
766 \newcommand*\includestandalone[2][[]]{%
767     \begingroup
768     \edef\@tempa{{#2\sa@graphicext}}%
769     \expandafter\includestandalone@\@tempa{#2}{#1}%
770     \endgroup
771 }
772
773 %% 0 = PDF if exists, TEX otherwise
774 %% 1 = force TEX
775 %% 2 = force PDF
776 %% 3 = build PDF if not exists
777 %% 4 = build PDF if older than TEX
778 \def\includestandalone@#1#2#3{%
779     \ifcase\sa@mode
780     \relax% 0
781         \IfFileExists{#1}%
782             {\includegraphics[#3]{#1}}%

```

```

781         {\includegraphics [#3]{#2.tex}}%
782 \or% 1
783     \includegraphics [#3]{#2.tex}%
784 \or% 2
785     \includegraphics [#3]{#1}%
786 \or% 3
787     \IfFileExists{#1}%
788         {\includegraphics [#3]{#1}}%
789         {\sa@buildgraphic{#2}}%
790         \IfFileExists{#1}%
791             {\includegraphics [#3]{#1}}%
792             {\PackageWarning{standalone}%
793             {Graphic '#1' could not be build.^^J%
794             Shell escape activated?}}%
795             \includegraphics [#3]{#2.tex}%
796         }%
797     }%
798 \else% 4
799     \IfFileExists{#1}%
800         {\sa@comparegraphic{#2}}%
801         {\sa@buildgraphic{#2}}%
802         \IfFileExists{#1}%
803             {\includegraphics [#3]{#1}}%
804             {\PackageWarning{standalone}%
805             {Graphic '#1' could not be build.^^J%
806             Shell escape activated?}}%
807             \includegraphics [#3]{#2.tex}%
808         }%
809     }%
810 \fi
811 }
812 \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.

```

813 \PassOptionsToPackage{active , tightpage}{preview}%
814

```

```
815 \ifstandalonebeamer
816   \setbeamertemplate{navigation symbols}{}%
817 \fi
```