Emulating Former KOMA-Script Package scrhack

Markus Kohm

Version 2024-09-04 v3.42

For several years KOMA-Script provided a package scrhack, that has been made not only to improve the compatibility of third-party packages with KOMA-Script classes, but also to generally improve third-party packages. The package consisted not only in the package file `scrhack.sty` but also in several so called *hacks* as separate files with extension `.hak`. In the course of splitting off former KOMA-Script packages from the KOMA-Script collection, it was decided to create independent packages from the individual hacks. These new independent packages then serve as replacements for the original packages they were intended to improve. In addition, to preserve the functionality of loading the enhancements only when the corresponding original package is loaded, a new scrhack was created to do just that.

Contents

1 History

Some packages from other authors do not work well with KOMA-Script. It is often very tedious for the author of KOMA-Script to convince the authors of these packages to make specific improvements. This also applies to packages whose development has been discontinued. That's why the scrhack was created. This package alters the commands and definitions of other to work better with KOMA-Script. Some changes are also useful when using other classes.

In the early days of KOMA-Script, users wanted to handle lists of floating environments created with the float package in the same way as the list of figures and list of tables created by KOMA-Script itself. At that time the author of KOMA-Script contacted the author of float to propose an interface to support such an enhancement. A somewhat modified form of that interface was implemented with the `\float@listhead` and `\float@addtolists` commands.

Later it became apparent that these two commands were not flexible enough to fully support all of KOMA-Script's capabilities. Unfortunately, the author of float had already ceased development by that point, so further changes to this package cannot be expected.

Other package authors have also adopted these two commands, and it became apparent that the implementation in some packages, including float, means that all these packages can only be loaded in a specific order, even though they are otherwise unrelated to each other.

To overcome all these disadvantages and problems, KOMA-Script no longer officially supports this old interface. Instead, KOMA-Script warns if the old interface is used. At the same time, the tocbasic package has been designed and implemented as a central interface for managing of table of contents and similar content lists. This package provides many more advantages and features than the two old commands.

Although the effort to use this package is very small, so far the authors of most of the packages that use the old interface have not made any adjustments. Therefore the KOMA-Scriptauthor designed a new package scrhack as part of KOMA-Script. That package contained appropriate modifications of the packages float, floatrow, listings, and later on also for nomencl and some others not related to that interface. Merely loading scrhack was sufficient to make these packages recognize not only the setting of the KOMA-Script option `listof`, but, e.g., also to react to the language switching features of the babel package. You can find more information about the features available by switching packages to tocbasic in [Koh23a] or [Koh23b].

With release of version 3.42 package scrhack has been removed from KOMA-Script. The code of the former package has been split into several standalone packages, that can be loaded individually if needed. However, for compatibility and maybe for convenience there is also a new standalone package scrhack almost compatible with the old one.

2 Using scrhack

It is recommended to load the scrhack package using:

```
\usepackage{scrhack}
```

as early as possible. In particular, loading should be done before the affected packages. The package also provides several options, one for each hack. Usually the options are named

identically to the package, that is patched by the hack. All these options are boolean options. You can switch such an option on assigning value `true` or using it without value. You can switch off such an option assigning value `false`. Switching off an option means to not automatically load the corresponding enhancement package after the package, that has to be patched. Therefore switching off an option results in not patching the corresponding package.

Notes:

- From version 3.42, you cannot use `\KOMAoptions` or `\KOMAoption` any longer to change the options. Instead loading scrhack again with other options does not result in an option clash, but in a change of the option settings.
- Switching off an option of an already loaded enhancement package is ignored (with warning). It does not unload the enhancement package or switch back to a not patched functionality.
- Switching on an option of a not yet loaded enhancement package would result in loading the enhancement package.
- From version 3.42, you should no longer load package scrhack. Instead you can and should simply load the individual enhancement package when you need it.
- If you need the old legacy version of scrhack from KOMA-Script 3.41 you can use:

  ```
  \usepackage{scrhack}[=2023-07-07]
  ```

  or

  ```
  \usepackage{scrhack}[=v3.41]
  ```

  However, this depends on internal packages of KOMA-Script.
- The hacks of the legacy version have been frozen. So the hacks may break with packages or LaTeX versions released after 2023-07-07. There is no support for such combinations!
- Loading the legacy version more than once with different option setting still results in option clash errors.

Available options with scrhack:

`float=`⟨*boolean*⟩: initial: `true`, default: `true`

After package float is loaded, load also package floatbytocbasic to fix the issue explained in section 1. This also provides the full functionality of tocbasic to float.

`floatrow=`⟨*boolean*⟩: initial: `true`, default: `true`

After package floatrow is loaded, load also package floatbytocbasic to fix the issue explained in section 1. This also provides the full functionality of tocbasic to floatrow.

`hyperref=`⟨*boolean*⟩: initial: `false`, default: `true`

Option `hyperref` is deprecated. It is implemented for compatibility reasons only. Using `hyperref` without value or `hyperref=true` results in a warning. Using `hyperref=false` results in an info only.

`listings=`⟨*boolean*⟩: initial: `false`, default: `true`

Option `listings` is deprecated. It is implemented for compatibility reasons only. Using `listings` without value or `listings=true` results in a warning. Using `listings=false` results in an info only.

`lscape=`⟨*boolean*⟩: initial: `true`, default: `true`

After package lscape is loaded, load also package lscapeenhanced to fix an issue with detection of text height for some packages like scrlayer and showframe.

`nomencl=`⟨*boolean*⟩: initial: `false`, default: `true`

Option `nomencl` is deprecated. It is implemented for compatibility reasons only. Using `nomencl` without value or `nomencl=true` results in a warning. Using `nomencl=false` results in an info only.

`setspace=`⟨*boolean*⟩: initial: `true`, default: `true`

After package setspace is loaded, load also package setspaceenhanced with the same options plus options `byselectfont` and `keepfontsize`.

`standardsectioning=`⟨*boolean*⟩: initial: `false`, default: `true`

Load package standardsectioning to (re-)define the sectioning commands using the code from the standard classes. If scrhack is loaded *before* any class[1] the definition is delayed until a class has been loaded.

Available options with the legacy version:

`float=`⟨*boolean*⟩: initial: `true`, default: `true`

After package float is loaded, load `float.hak` to fix the issue explained in section 1. This also provides the full functionality of tocbasic to float.

`floatrow=`⟨*boolean*⟩: initial: `true`, default: `true`

After package floatrow is loaded, load `floatrow.hak` to fix the issue explained in section 1. This also provides the full functionality of tocbasic to floatrow.

`hyperref=`⟨*boolean*⟩: initial: `true`, default: `true`

Versions of hyperref before 6.79h set the link anchors after the heading of the starred versions of commands like `\part*`, `\chapter*`, etc. instead of before them. Since then, this problem has been resolved at the suggestion of KOMA-Script's author. But because the change took more than a year, a patch was added to scrhack. Although this can also be deactivated with `hyperref=false`, you should instead use an up-to-date hyperref release. In this case the legacy version of scrhack automatically deactivates this unnecessary patch.

[1] This is detected heuristic by definition of `\normalsize`. If `\normalsize` contains `\@latex@error` no class has been loaded.

`listings=`⟨*boolean*⟩: initial: `true`, default: `true`

After package listings is loaded, load `listings.hak` to fix the issue explained in section 1. This also provides the full functionality of tocbasic to old versions of listings. With newer versions, also the legacy version of scrhack automatically deactivates this unnecessary patch.

`lscape=`⟨*boolean*⟩: initial: `true`, default: `true`

The lscape package defines a `landscape` environment to set the page contents, but not the header or footer in landscape mode. Inside this environment, `\textheight`is set to the value of `\textwidth`, but `\textwidth` is not set to the former value of `\textheight`. This is inconsistent. As far as I know, `\textwidth` is left unchanged because setting it to `\textheight` could interfere with other packages or user commands. But changing `\textheight` also has this potential, and indeed it breaks, for example, showframe and scrlayer. Thus it would be best if `\textheight` too remained unchanged. After package lscape is loaded, the legacy version of scrhack loads `lscape.hak`, that uses the xpatch package to modify the `landscape` environment's start macro `\landscape` appropriately.

Incidentally, the pdflscape package also uses lscape, so this option affects the functioning of this package too.

`nomencl=`⟨*boolean*⟩: initial: `true`, default: `true`

After package nomencl is loaded, load `nomencl.hak` to fix the issue explained in section 1. This is a little bit more complicated, because the nomencl author has also adapted his package to support tocbasic. So additional versions checks are done.

`setspace=`⟨*boolean*⟩: initial: `true`, default: `true`

Some packages assume that the class-internal macro `\@ptsize` both is defined and expands to an integer. For compatibility, KOMA-Script defines `\@ptsize` even if the basic font size is something other than 10 pt, 11 pt, or 12 pt. KOMA-Script also allows non-integer font sizes. So `\@ptsize` can, of course, also expand to a non-integer number.

One of the packages that cannot cope with a non-integer `\@ptsize` is setspace. Additionally, the values set by this package are always dependent on the basic font size, even if the setting is made in the context of another font size. The legacy version of scrhack loads the hack `setspace.hak` after package setspace to solves both problems by redefining `\onehalfspacing` and `\doublespacing` to set the spacing relative to the actual font size.

Note: if you use setspace with either the `onehalfspacing` or `doublespacing` option, you must load scrhack first.

`standardsections=`⟨*boolean*⟩ initial: `false`, default: `true`

Various packages assume that the sectioning commands are defined in a specific way, corresponding to the definitions in the standard classes. But for some classes this is not the case. For example, the KOMA-Script classes use a completely different implementation to provide many additional features. But this can cause problems for packages that depend on the definition of the standard classes. The legacy version of scrhack offers the option to force the sectioning commands `\part`, `\chapter`, `\section`, `\subsection`, `\subsubsection`, `\paragraph`, and `\subparagraph` to be compatible

with those in the standard classes. When `\chapter` is defined, the definitions are based on those in book. When `\chapter` is undefined, the definitions of article are used.

If you are using a KOMA-Script class, several features of these classes are also deactivated as side effect. For example, the commands to define or modify sectioning commands or option `headings` are no longer available, and commands like `\partformat` have different defaults.

Because this hack has the potential to do more harm than good, it issues several warnings. Also it is not activated simply by loading the legacy version of scrhack package. If you want to use it, you must explicitly activate this option when you load the package. Late activation or deactivation is not supported.

Since there are often less invasive solutions to fix the problem of package incompatibilities, using this hack is not recommended. It is provided only as a last resort for emergencies.

3 Implementation

Because scrhack is not longer a KOMA-Script package, it does not use the `KOMA` family for options any longer. Instead it uses the new LaTeX kernel feature of key-value-options introduced in [TLT22]. So we need at least LaTeX 2022-06-01:

```
\ifnum 0=\ifcsname IfFormatAtLeastTF\endcsname
    \IfFormatAtLeastTF{2022-06-01}{1}{0}%
  \else
    0%
  \fi\relax
  \PackageError{scrhack}{LaTeX kernel too old}{%
    The package needs at least LaTeX 2022-06-01.\MessageBreak
    This error is fatal. Loading will be aborted.%
  }%
  \endinput
\fi
\ExplSyntaxOn
```

3.1 Messages

We need some messages, that are not specific for only one use-case.

- If an option is ignored, because it cannot be handled after already loading the enhancement package. #1 is the name option, #2 is the enhancement package.

```
\msg_new:nnn { scrhack } { option-too-late }
  {
    option~‘#1’~after~loading~package~‘#2’~ignored.
  }
```

- If an option is ignored, because it is deprecated.

```
\msg_new:nnn { scrhack } { deprecated-option }
  {
```

```
    deprecated~option~‘#1’~ignored.
  }
```

3.2 Emulating `float.hak`

First we declare the options using the new key-value interface:

```
\DeclareKeys{%
  float .code      = \@@_switch_float:n { #1 },
  float .default:n = true,
  float .usage     = preamble,
}
```

`\__scrhack_switch_float:n` This is used to switch the state and add or remove the hook if needed. A new boolean is used to know, whether the hook has been put. Using `float=true`, if package floatbytocbasic already has been loaded isn't a foul. So we just write an information to the `log`. But using `float=false`, if package floatbytocbasic already has been loaded may be wrong. So we use a warning message. We also warn if a not allowed value is used.

```
\bool_new:N \@@_float_bool

\cs_new:Nn \@@_switch_float:n
  {
    \str_case_e:nnF { \str_foldcase:n { #1 } }
      {
        {true}  {
                  \@ifpackageloaded { floatbytocbasic }
                    {
                      \msg_info:nnnn { scrhack } { option-too-late }
                        { float = true } { floatbytocbasic }
                    }
                    {
                      \bool_if:NF \@@_float_bool
                        {
                          \hook_gput_code:nnn { package/float/after } { scrhack }
                            {
                              \RequirePackage { floatbytocbasic }
                            }
                          \bool_gset_true:N \@@_float_bool
                        }
                    }
                }
        {false} {
                  \@ifpackageloaded { floatbytocbasic }
                    {
                      \msg_warning:nnnn { scrhack } { option-too-late }
                        { float=false } { floatbytocbasic }
                    }
                    {
                      \bool_if:NT \@@_float_bool
                        {
                          \hook_gremove_code:nn { package/float/after } { scrhack }
                        }
```

```
                    \bool_gset_false:N \@s@_float_bool
                  }
              }
      }
      {
        \msg_warning:nnn { keys } { boolean-values-only } { float }
      }
  }
\SetKeys{float=true}
```

3.3 Emulating `floatrow.hak`

First we declare the options using the new key-value interface:

```
\DeclareKeys{%
  floatrow .code      = \@@_switch_floatrow:n { #1 },
  floatrow .default:n = true,
  floatrow .usage     = preamble,
}
```

`\__scrhack_switch_floatrow:n` This is used to switch the state and add or remove the hook if needed. A new boolean is used to know, whether the hook has been put. Using `floatrow=true`, if package floatrowbytocbasic already has been loaded isn't a foul. So we just write an information to the `log`. But using `floatrow=false`, if package floatrowbytocbasic already has been loaded may be wrong. So we use a warning message. We also warn if a not allowed value is used.

```
\bool_new:N \@@_floatrow_bool

\cs_new:Nn \@@_switch_floatrow:n
  {
    \str_case_e:nnF { \str_foldcase:n { #1 } }
      {
        {true}  {
                  \@ifpackageloaded { floatrowbytocbasic }
                    {
                      \msg_info:nnnn { scrhack } { option-too-late }
                        { floatrow = true } { floatrowbytocbasic }
                    }
                    {
                      \bool_if:NF \@@_floatrow_bool
                        {
                          \hook_gput_code:nnn { package/floatrow/after }
                                              { scrhack }
                            {
                              \RequirePackage { floatrowbytocbasic }
                            }
                          \bool_gset_true:N \@@_floatrow_bool
                        }
                    }
                }
        {false} {
                  \@ifpackageloaded { floatrowbytocbasic }
                    {
```

```
                    \msg_warning:nnnn { scrhack } { option-too-late }
                      { floatrow=false } { floatrowbytocbasic }
                  }
                  {
                    \bool_if:NT \@@_floatrow_bool
                      {
                        \hook_gremove_code:nn { package/floatrow/after }
                                              { scrhack }
                      }
                    \bool_gset_false:N \@s@_floatrow_bool
                  }
              }
      }
      {
        \msg_warning:nnn { keys } { boolean-values-only } { floatrow }
      }
  }
\SetKeys{floatrow=true}
```

3.4 Emulating `hyperref.hak`

First we declare the options using the new key-value interface:

```
\DeclareKeys{%
  hyperref .code      = \@@_switch_hyperref:n { #1 },
  hyperref .default:n = true,
  hyperref .usage     = preamble,
}
```

`\__scrhack_switch_hyperref:n` This is somehow special, because the hack is deprecated. It only has been needed for hyperref versions released before 2009-11-24, which would not work with the LaTeX kernel version needed by this version of scrhack. So emulating the hack does not make sense. Instead the initial value of the option is changed to implicit `false` and switching it on explicitly does result in a warning.

```
\cs_new:Nn \@@_switch_hyperref:n
  {
    \str_case_e:nnF { \str_foldcase:n { #1 } }
      {
        {true}  {
                  \msg_warning:nnn { scrhack } { deprecated-option }
                                   { hyperref=#1 }
                }
        {false} {
                  \msg_info:nnn { scrhack } { deprecated-option }
                                { hyperref=#1 }
                }
      }
      {
        \msg_warning:nnn { keys } { boolean-values-only } { hyperref }
      }
  }
```

3.5 Emulating `listings.hak`

First we declare the options using the new key-value interface:

```
\DeclareKeys{%
  listings .code      = \@@_switch_listings:n { #1 },
  listings .default:n = true,
  listings .usage     = preamble,
}
```

`\__scrhack_switch_listings:n` This is somehow special, because the hack is deprecated. It only has been needed for listings version released before 2024-02-15. So instead of using any patch hacks, users should update listings. So emulating the hack does not make sense. Instead the initial value of the option is changed to implicit `false` and switching it on explicitly does result in a warning.

```
\cs_new:Nn \@@_switch_listings:n
  {
    \str_case_e:nnF { \str_foldcase:n { #1 } }
      {
        {true}  {
                  \msg_warning:nnn { scrhack } { deprecated-option }
                                   { listings=#1 }
                }
        {false} {
                  \msg_info:nnn { scrhack } { deprecated-option }
                                { listings=#1 }
                }
      }
      {
        \msg_warning:nnn { keys } { boolean-values-only } { listings }
      }
  }
```

3.6 Emulating `lscape.hak`

First we declare the options using the new key-value interface:

```
\DeclareKeys{%
  lscape .code      = \@@_switch_lscape:n { #1 },
  lscape .default:n = true,
  lscape .usage     = preamble,
}
```

`\__scrhack_switch_lscape:n` This is used to switch the state and add or remove the hook if needed. A new boolean is used to know, whether the hook has been put. Using `lscape=true`, if package lscapeenhanced already has been loaded isn't a foul. So we just write an information to the `log`. But using `lscape=false`, if package lscapeenhanced already has been loaded may be wrong. So we use a warning message. We also warn if a not allowed value is used.

```
\bool_new:N \@@_lscape_bool

\cs_new:Nn \@@_switch_lscape:n
  {
    \str_case_e:nnF { \str_foldcase:n { #1 } }
      {
```

```
        {true}  {
                  \@ifpackageloaded { lscapeenhanced }
                    {
                      \msg_info:nnnn { scrhack } { option-too-late }
                        { lscape = true } { lscapeenhanced }
                    }
                    {
                      \bool_if:NF \@@_lscape_bool
                        {
                          \hook_gput_code:nnn { package/lscape/after }
                                              { scrhack }
                            {
                              \RequirePackage { lscapeenhanced }
                            }
                          \bool_gset_true:N \@@_lscape_bool
                        }
                    }
                }
        {false} {
                  \@ifpackageloaded { lscapeenhanced }
                    {
                      \msg_warning:nnnn { scrhack } { option-too-late }
                        { lscape=false } { lscapeenhanced }
                    }
                    {
                      \bool_if:NT \@@_lscape_bool
                        {
                          \hook_gremove_code:nn { package/lscape/after }
                                                { scrhack }
                        }
                      \bool_gset_false:N \@s@_lscape_bool
                    }
                }
      }
      {
        \msg_warning:nnn { keys } { boolean-values-only } { lscape }
      }
  }
\SetKeys{lscape=true}
```

3.7 Emulating `nomencl.hak`

First we declare the options using the new key-value interface:

```
\DeclareKeys{%
  nomencl .code      = \@@_switch_nomencl:n { #1 },
  nomencl .default:n = true,
  nomencl .usage     = preamble,
}
```

`\__scrhack_switch_nomencl:n` This is somehow special, because the hack is deprecated. It only has been needed for nomencl versions released before 2019-01-23. So updating the package would be the better *hack*. So

emulating the hack does not make sense. Instead the initial value of the option is changed to implicit `false` and switching it on explicitly does result in a warning.

```
\cs_new:Nn \@@_switch_nomencl:n
  {
    \str_case_e:nnF { \str_foldcase:n { #1 } }
      {
        {true}  {
                  \msg_warning:nnn { scrhack } { deprecated-option }
                                   { nomencl=#1 }
                }
        {false} {
                  \msg_info:nnn { scrhack } { deprecated-option }
                                { nomencl=#1 }
                }
      }
      {
        \msg_warning:nnn { keys } { boolean-values-only } { nomencl }
      }
  }
```

3.8 Emulating `setspace.hak`

First we declare the options using the new key-value interface:

```
\DeclareKeys{%
  setspace .code      = \@@_switch_setspace:n { #1 },
  setspace .default:n = true,
  setspace .usage     = preamble,
}
```

`\__scrhack_switch_setspace:n` This is used to switch the state and add or remove the hook if needed. A new boolean is used to know, whether the hook has been put. Using `setspace=true`, if package setspaceenhanced already has been loaded isn't a foul. So we just write an information to the `log`. But using `setspace=false`, if package setspaceenhanced already has been loaded may be wrong. So we use a warning message. We also warn if a not allowed value is used.

```
\bool_new:N \@@_setspace_bool

\cs_new:Nn \@@_switch_setspace:n
  {
    \str_case_e:nnF { \str_foldcase:n { #1 } }
      {
        {true}  {
                  \@ifpackageloaded { setspaceenhanced }
                    {
                      \msg_info:nnnn { scrhack } { option-too-late }
                        { setspace = true } { setspaceenhanced }
                    }
                    {
                      \bool_if:NF \@@_setspace_bool
                        {
                          \hook_gput_code:nnn { package/setspace/after } { scrhack }
                            {
```

```
                                \exp_last_unbraced:Ne \RequirePackage
                                  {
                                    [ \use:c { @raw@opt@setspace.sty },
                                      byselectfont,
                                      keepfontsize ]
                                  }
                                  { setspaceenhanced }
                              }
                            \bool_gset_true:N \@@_setspace_bool
                          }
                      }
                  }
        {false} {
                    \@ifpackageloaded { setspaceenhanced }
                      {
                        \msg_warning:nnnn { scrhack } { option-too-late }
                          { setspace=false } { setspaceenhanced }
                      }
                      {
                        \bool_if:NT \@@_setspace_bool
                          {
                            \hook_gremove_code:nn { package/setspace/after } { scrhack }
                          }
                        \bool_gset_false:N \@s@_setspace_bool
                      }
                  }
      }
      {
        \msg_warning:nnn { keys } { boolean-values-only } { setspace }
      }
  }
\SetKeys{setspace=true}
```

3.9 Emulating `standardsectioning.hak`

First we declare the options using the new key-value interface:

```
\DeclareKeys{%
  standardsections .code      = \@@_switch_standardsectioning:n { #1 },
  standardsections .default:n = true,
  standardsections .usage     = preamble,
}
```

`ack_switch_standardsectioning:n` This is used to switch the state and add or remove the hook if needed. A new boolean is used to know, whether the hook has been put. Using `standardsectioning=true`, if package standardsectioning already has been loaded isn't a foul. So we just write an information to the `log`. But using `standardsectioning=false`, if package standardsectioning already has been loaded may be wrong. So we use a warning message. We also warn if a not allowed value is used.

```
\bool_new:N \@@_standardsectioning_bool

\cs_new:Nn \@@_switch_standardsectioning:n
```

```
{
  \str_case_e:nnF { \str_foldcase:n { #1 } }
    {
      {true}  {
                \@ifpackageloaded { standardsectioning }
                  {
                    \msg_info:nnnn { scrhack } { option-too-late }
                      { standardsectioning = true } { standardsectioning }
                  }
                  {
                    \bool_if:NF \@@_standardsectioning_bool
                      {
                        \str_set:Nx \l_tmpa_str { \cs_meaning:N \normalsize }
                        \str_if_in:NnTF
                          \l_tmpa_str
                          { \@latex@error }
                          {
                            \hook_gput_code:nnn { class/after } { scrhack }
                              {
                                \RequirePackage { standardsectioning }
                                \exp_args:Nnnx \hook_gput_code:nnn
                                  { begindocument/before } { scrhack }
                                  {
                                    \exp_not:N \cs_gset:Npn
                                    \exp_after:wN \exp_not:N
                                    \cs:w ver@standardsectioning.sty \cs_end:
                                      {
                                        \cs:w ver@standardsectioning.sty \cs_end:
                                      }
                                  }
                                \cs_undefine:c { var@standardsectioning.sty }
                              }
                            \bool_gset_true:N \@@_standardsectioning_bool
                          }
                          {
                            \RequirePackage{ standardsectioning }
                            \bool_gset_false:N \@@_standardsectioning_bool
                          }
                      }
                  }
              }
      {false} {
                \@ifpackageloaded { standardsectioning }
                  {
                    \msg_warning:nnnn { scrhack } { option-too-late }
                      { standardsectioning=false } { standardsectioning }
                  }
                  {
                    \bool_if:NT \@@_standardsectioning_bool
                      {
                        \hook_gremove_code:nn { class/after } { scrhack }
                      }
```

```
                    \bool_gset_false:N \@s@_standardsectioning_bool
                  }
              }
    }
    {
      \msg_warning:nnn { keys } { boolean-values-only } { standardsectioning }
    }
}
```

3.10 Processing the options

Last but not least, we set the defaults and process the options.

```
\ExplSyntaxOff
\ProcessKeyOptions\relax
```

References

[Koh23a] Markus Kohm. *KOMA-Script. The Guide.* June 16, 2023. URL: http://mirrors.ctan.org/macros/latex/contrib/koma-script/scrguide-en.pdf (visited on 07/14/2023).

[Koh23b] Markus Kohm. *KOMA-Script. Die Anleitung.* June 16, 2023. URL: http://mirrors.ctan.org/macros/latex/contrib/koma-script/scrguide-de.pdf (visited on 07/04/2023).

[Koh23c] Markus Kohm. *KOMA-Script — A bundle of versatile classes and packages.* Version 3.41. The KOMA-Script bundle provides replacements for the article, report, and book classes with emphasis on typography and versatility. There is also a letter class. July 7, 2023. URL: https://ctan.org/pkg/koma-script (visited on 07/14/2023).

[TLT22] The LaTeX Project Team. "Issue 35." In: *LaTeX News* (June 2022). URL: http://mirrors.ctan.org/macros/latex/base/ltnews35.pdf (visited on 07/14/2023).

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.

Change History

v0.1 – 2023/07/14
General: start of KOMA-Script spin-off 1