KOMA-Script File `scrhack.dtx` *

Markus Kohm†

package

Some packages from other authors may have problems with KOMA-Script. In my opinion some packages could be improved. With some packages this makes only sense, if KOMA-Script was used. With some other packages the package author has another opinion. Sometimes proposals was never answered. Package scrhack contains all those improvement proposals for other packages. This means, scrhack redefines macros of packages from other authors! The redefinitions are only activated, if those packages were loaded. Users may prevent scrhack from redefining macros of individual packages.

Contents

*This file is version (hacking of `scrhack.dtx`.

†mailto:komascript(at)gmx.info

1 The hyperref hack

Before version 6.79h package hyperref does behave different at part, chapter, and section headings that get no number. If they get no number, because of to low counter

`secnumdepth`

hyperref sets an anchor for links and bookmarks before the heading. Same would be, if the headings have a number. But if the headings get no number because of usage of the star version of the commands, e.g., `\part*`, `\chapter*` or `\section*`, the anchor for links and bookmarks are set after the headings. The anchors for numbered headings are always set before the headings.

Package scrhack redefines some macros of some hyperref driver files, e.g., hpdftex.def, after loading the hyperref driver file. With this redefinitions the anchor of not numbered headings will be set always before the headings, too.

You may switch off the hyperref hack loading package scrhack with option `hyperref=false`. You may also switch off the hyperref hack using `\KOMAoptions{hyperref=false}` or `\KOMAoption{hyperref}{false}` somewhere after loading package scrhack, but before loading the hyperref driver package, that is by default after loading the package.

2 The float hack

Package float uses macros `\float@listhead` to set the headings of a float listing and `\float@addtolists` to add informations to all float listings. These macros where proposed by the KOMA-Script author for some years. In theory those macros may be used by several class and package authors to deligate some parts of the creation of a float listing to the class. This would increase the compatiblity of packages and classes. But unfortunately some package authors, even the author of package float, implemented the commands in such a way, that these packages will become incompatible to each other.

Because of this KOMA-Script stopped support for `\float@addtolists` and `\float@listhead` with version 3. Instead of this KOMA-Script supports several improvements for package authors using KOMA-Script package tocbasic.

Package scrhack redefines some macros of package float to not longer use `\float@addtolists` and `\float@listhead` but use the interface of package tocbasic. This does not only improve the compatibility of KOMA-Script and package float, but also improves the compatibility of packages babel and float.

You may switch off the float hack loading package scrhack with option `float=false`. You may also switch off the float hack using `\KOMAoptions{float=false}` or `\KOMAoption{float}{false}` somewhere after loading package scrhack, but before loading package float.

3 The floatrow hack

Package floatrow uses macros `\float@listhead` to set the headings of a float listing and `\float@addtolists` to add informations to all float listings. These macros where proposed by the KOMA-Script author for some years. In theory those macros may be used by several class and package authors to deligate some parts of the creation of a float listing to the class. This would increase the compatiblity of packages and classes. But unfortunately some package authors, even the author of package floatrow, implemented the commands in such a way, that these packages will become incompatible to each other.

Because of this KOMA-Script stopped support for `\float@addtolists` and `\float@listhead` with version 3. Instead of this KOMA-Script supports several improvements for package authors using KOMA-Script package tocbasic.

Package scrhack redefines some macros of package floatrow to not longer use `\float@addtolists` and `\float@listhead` but use the interface of package tocbasic. This does not only improve the compatibility of KOMA-Script and package floatrow, but also improves the compatibility of packages babel and floatrow.

You may switch off the floatrow hack loading package scrhack with option `floatrow=false`. You may also switch off the floatrow hack using `\KOMAoptions{floatrow=false}` or `\KOMAoption{floatrow}{false}` somewhere after loading package scrhack, but before loading package floatrow.

4 The listings hack

Package listings uses macros `\float@listhead` to set the headings of a float listing, if defined, and `\float@addtolists` to add informations to all float listings. These macros where proposed by the KOMA-Script author for some years. In theory those macros may be used by several class and package authors to deligate some parts of the creation of a float listing to the class. This would increase the compatiblity of packages and classes. But unfortunately some package authors, even the author of package float, impemented the commands in such a way, that these packages may become incompatible to each other.

Because of this KOMA-Script stopped support for `\float@addtolists` and `\float@listhead` with version 3. Instead of this KOMA-Script supports several improvements for package authors using KOMA-Script package tocbasic.

Package scrhack redefines some macros of package listings to not longer use `\float@addtolists` and `\float@listhead` but use the interface of package tocbasic. This does not only improve the compatibility of KOMA-Script and package listings, but also improves the compatibility of packages babel and listings.

Note: A significant change with scrhack is, that KOMA-Script options like `lists=totoc` or `lists=totocnumbered` does only change the behaviour of `\listoflistings`, if they are set after loading package listings.

You may switch off the listings hack loading package scrhack with option `listings=false`. You may also switch off the listings hack using `\KOMAoptions{listings=false}` or `\KOMAoption{listings}{false}` somewhere after loading package scrhack, but before loading package listings.

5 The setspace hack

Package setspace defines macros `\onehalfspacing` and `\doublespacing` using `\@ptsize` as an argument of `\ifcase`. But if `\@ptsize` is not an integer but a real number, this failes, because the digits from the decimal points are interpreted as text of that case. Several solutions for this are thinkable. I've decides to redefine `\onehalfspacing` and `\doublespacing`. The new definition is more general and somehow more exact.

You can switch of the setspace hack loading package scrhack with option `setspace=false`. You may also switch of the setspace hack using `\KOMAoptions{setspace=false}` or `\KOMAoption{setspace}{false}` somewhere after loading package scrhack, but before loading package setspace.

Note: If you want to use setspace with package option `onehalfspacing` or `doublespacing` you have to load scrhack before setspace.

6 The lscape hack

Package lscape defines an environment `landscape` to set the page contents but not head and foot landscape. Inside this environment it changes `\textheight` to the value of `\textwidth`, but it does not change `\textwidth` to the former value of `\textheight`. This is inconsistent. As far a I know, `\textwidth` is unchanged because setting it to `\textheight` could blame other packages or user commands. But changing `\textheight` could also blame other packages or user commands and indeed it breaks, e. g., showframe and scrlayer. So best would be, not to change `\textheight`, too. scrhack uses package xpatch to modify the environment start macro `\landscape` appropriately.

You can switch of the lscape hack loading package scrhack with option `lscape=false`. You can also change option `lscape` afterwards. If the option is `false` while loading lscape, scrhack will not patch `\landscape` and later changes of the option have no effect. But if the option is `true` while loading lscape or if scrhack is loaded after lscape without option `lscape=false`, every later change of the option using `\KOMAoption` or `\KOMAoptions` will have the expected effect.

7 Implementation of scrhack

7.1 Optionen

Das Paket bedient sich `\KOMAoptions` etc. aus scrkbase (dieses wird übrigens direkt per `scrkbase.dtx` geladen).

Per Option kann gewählt werden, welche Manipulationen geladen werden sollen. Alle diese Optionen können jedoch nur bis zum Laden des entsprechenden Pakets oder dem Laden von scrhack gesetzt werden (es zählt, was später kommt). Anschließend sind sie wirkungslos.

7.2 Verwendete Anweisungen

`\scr@ifexpected` Wenn die im ersten Argument angegebene Anweisung nach Ausführung der im zweiten Argument angegebenen Anweisungen unverändert ist, dann soll das dritte Argument ausgeführt werden, sonst das vierte.

```
\newcommand{\scr@ifexpected}[2]{%
  \begingroup
    \let\@tempa#1
    #2
    \ifx\@tempa#1
      \aftergroup\@firstoftwo
    \else
      \aftergroup\@secondoftwo
    \fi
  \endgroup
}
```

`\scr@hack@load` Wenn die Datei mit dem Namen des zweiten Arguments und der Endung des ersten Arguments so geladen wurde, dass LaTeX eine Versionsinfo dazu gespeichert hat, dann soll zusätzlich der entsprechende Hack geladen werden.

```
\newcommand*{\scr@hack@load}[2]{%
  \expandafter\ifx\csname ver@#2.#1\endcsname\relax
    \expandafter\@secondoftwo
  \else
```

Allerdings wird jeder Hack nur genau einmal geladen:

```
    \expandafter\ifx\csname ver@#2.hak\endcsname\relax
      \expandafter\expandafter\expandafter\@firstoftwo
    \else
      \expandafter\expandafter\expandafter\@secondoftwo
    \fi
  \fi
  {%
    \PackageInfo{scrhack}{loading #2 hack}%
    \edef\reserved@a{%
      \noexpand\makeatletter\noexpand\input{#2.hak}%
      \noexpand\catcode`\noexpand\@\the\catcode`\@\relax
    }\reserved@a
  }{%
    \PackageInfo{scrhack}{ignorring #2 hack}%
  }%
}
```

7.3 Der hyperref-Hack

hyperref setzt den Anker zu der Stern-Variante einer Überschrift hinter die Überschrift, während es bei der nicht Stern-Variante den Anker auch dann vor die Überschrift setzt, wenn die Überschrift aufgrund von `secnumdepth` nicht nummeriert wird. Der Hack setzt den Anker einheitlich vor die Überschrift.

`hyperref`

```
⟨*package & option⟩
\KOMA@ifkey{hyperref}{@scrhack@hyperref}%
\KOMAExecuteOptions{hyperref=true}%
⟨/package & option⟩
⟨*package & body⟩
```

Hier muss ein wenig trickreicher gearbeitet werden, weil hyperref die Treiberdatei per `\AtEndOfPackage` lädt und der Hack erst danach installiert werden darf. Mit `\AfterPackage*` alleine, würde der Hack aber vor dem Laden der Treiberdatei installiert. Dafür können wir aber sicher sein, dass ein innerhalb von `\AfterPackage*` aufgerufenes `\AtEndOfPackage` garantiert nach dem Laden der Treiberdatei ausgeführt wird. Das funktioniert auch noch, wenn hyperref bereits geladen wurde. In dem Fall wird der Code einfach nach dem Ende von scrhack statt nach dem Ende von hyperref ausgeführt.

```
\BeforePackage{hyperref}{%
  \scr@ifundefinedorrelax{hy@insteadofrefstepcounter}{}{%
    \PackageInfo{scrhack}{hyperref hack deactivated because of\MessageBreak
      detection of KOMA-Script class, that doesn't\MessageBreak
      need that hack,}%
    \KOMAExecuteOptions[.scrhack.sty]{hyperref=false}%
  }%
}
\AfterPackage*{hyperref}{%
  \if@scrhack@hyperref
    \@ifpackagelater{hyperref}{2009/11/24}{%
      \PackageInfo{scrhack}{hyperref hack deactivated because of\MessageBreak
        detection of hyperref version, that doesn't\MessageBreak
        need that hack,}%
      \KOMAExecuteOptions[.scrhack.sty]{hyperref=false}%
    }{%
      \AtEndOfPackage{%
        \KOMA@key[.scrhack.sty]{hyperref}{%
          \PackageWarning{scrhack}{option `hyperref=#1' ignored}%
          \FamilyKeyStateProcessed
        }%
```

```
        \if@scrhack@hyperref\scr@hack@load\@pkgextension{hyperref}\fi
      }%
    }%
  \fi
}
⟨/package & body⟩
```

\@schapter
\@spart
\@ssect

Eigentlich wird hier gar nicht `hyperref.sty` verändert, sondern diverse Treiberdateien. Sobald das Paket hyperref geladen ist, ist auch die passende Treiberdatei geladen und außerdem sind alle Treiberdateien, die entsprechende Definitionen vornehmen, gleichermaßen betroffen. Also kann der entsprechende Patch einfach erfolgen, wenn hyperref geladen ist (was bereits von `\scr@hack@load` getestet wurde). Es muss also nur noch sichergestellt werden, dass die umzudefinierenden Macros derzeit den erwarteten Inhalt haben.

```
⟨*hyperref & body⟩
\scr@ifexpected\@schapter{%
  \def\@schapter#1{%
    \H@old@schapter{#1}%
    \begingroup
      \let\@mkboth\@gobbletwo
      \Hy@GlobalStepCount\Hy@linkcounter
      \xdef\@currentHref{\Hy@chapapp*.\the\Hy@linkcounter}%
      \Hy@raisedlink{%
        \hyper@anchorstart{\@currentHref}\hyper@anchorend
      }%
    \endgroup
  }%
}{%
  \PackageInfo{scrhack}{redefining \string\@schapter}%
  \def\@schapter#1{%
    \begingroup
      \let\@mkboth\@gobbletwo
      \Hy@GlobalStepCount\Hy@linkcounter
      \xdef\@currentHref{\Hy@chapapp*.\the\Hy@linkcounter}%
      \Hy@raisedlink{%
        \hyper@anchorstart{\@currentHref}\hyper@anchorend
      }%
    \endgroup
    \H@old@schapter{#1}%
  }%
}{%
  \scr@ifexpected\@schapter{%
    \def\@schapter#1{%
      \begingroup
        \let\@mkboth\@gobbletwo
        \Hy@GlobalStepCount\Hy@linkcounter
        \xdef\@currentHref{\Hy@chapapp*.\the\Hy@linkcounter}%
```

```
        \Hy@raisedlink{%
          \hyper@anchorstart{\@currentHref}\hyper@anchorend
        }%
      \endgroup
      \H@old@schapter{#1}%
    }%
  }{}{%
    \PackageWarningNoLine{scrhack}{unknown \string\@schapter\space
      definition found!\MessageBreak
      Maybe you are using a unsupported hyperref version}%
  }%
}

\scr@ifexpected\@spart{%
  \def\@spart#1{%
    \H@old@spart{#1}%
    \Hy@GlobalStepCount\Hy@linkcounter
    \xdef\@currentHref{part*.\the\Hy@linkcounter}%
    \Hy@raisedlink{%
      \hyper@anchorstart{\@currentHref}\hyper@anchorend
    }%
  }%
}{%
  \PackageInfo{scrhack}{redefining \string\@spart}%
  \def\@spart#1{%
    \Hy@GlobalStepCount\Hy@linkcounter
    \xdef\@currentHref{part*.\the\Hy@linkcounter}%
    \Hy@raisedlink{%
      \hyper@anchorstart{\@currentHref}\hyper@anchorend
    }%
    \H@old@spart{#1}%
  }%
}{%
  \scr@ifexpected\@spart{%
    \def\@spart#1{%
      \Hy@GlobalStepCount\Hy@linkcounter
      \xdef\@currentHref{part*.\the\Hy@linkcounter}%
      \Hy@raisedlink{%
        \hyper@anchorstart{\@currentHref}\hyper@anchorend
      }%
      \H@old@spart{#1}%
    }%
  }{}{%
    \PackageWarningNoLine{scrhack}{unknown \string\@spart\space
      definition found!\MessageBreak
      Maybe you are using a unsupported hyperref version}%
  }%
}

```

```
\scr@ifexpected\@ssect{%
  \def\@ssect#1#2#3#4#5{%
    \H@old@ssect{#1}{#2}{#3}{#4}{#5}%
    \phantomsection
  }%
}{%
  \PackageInfo{scrhack}{redefining \string\@ssect}%
  \def\@ssect#1#2#3#4#5{%
    \H@old@ssect{#1}{#2}{#3}{#4}{\phantomsection\ignorespaces#5}%
  }%
}{%
  \scr@ifexpected\@ssect{%
    \def\@ssect#1#2#3#4#5{%
      \H@old@ssect{#1}{#2}{#3}{#4}{\phantomsection\ignorespaces#5}%
    }%
  }{}{%
    \PackageWarningNoLine{scrhack}{unknown \string\@ssect\space
      definition found!\MessageBreak
      Maybe you are using a unsupported hyperref version}%
  }%
}
⟨/hyperref & body⟩
```

7.4 Der float-Hack

Das float-Paket verwendet das Makro `\float@listhead` zum Setzen der Überschriften. Dies wird seit KOMA-Script 3 nicht mehr empfohlen und fliegt demnächst komplett aus der Unterstützung. Stattdessen wird empfohlen, dass Pakete tocbasic unterstützen. Der Aufwand dafür ist sehr gering und wird mit vielen neuen Möglichkeiten belohnt.

Dieser Hack rüstet die tocbasic-Unterstützung für float nach.

`float`

```
⟨*package & option⟩
\KOMA@ifkey{float}{@scrhack@float}%
\KOMAExecuteOptions{float=true}%
⟨/package & option⟩
⟨*package & body⟩
\AfterPackage*{float}{%
  \KOMA@key[.scrhack.sty]{float}{%
    \PackageWarning{scrhack}{option ‘float’ ignored}%
    \FamilyKeyStateProcessed
  }%
  \if@scrhack@float\scr@hack@load\@pkgextension{float}\fi
}
⟨/package & body⟩
```

`\newfloat` Über die Anweisung `\newfloat` wird eine neue Gleitumgebung definiert. Hier muss die neue Erweiterung aus dem dritten Argument tocbasic bekannt gemacht werden.

`\listof` Über die Anweisung `\listof` wird ein Verzeichnis für Gleitumgebungen ausgegeben. Hier muss schlicht die entsprechende Anweisung von tocbasic verwendet werden.

`\float@addtolists` Diese Anweisung wird nicht länger benötigt und daher auf die ursprüngliche Definition zurückgesetzt.

```
⟨∗float & body⟩
\scr@ifexpected{\newfloat}{%
  \long\def\newfloat#1#2#3{\@namedef{ext@#1}{#3}
    \let\float@do=\relax
    \xdef\@tempa{\noexpand\float@exts{\the\float@exts \float@do{#3}}}%
    \@tempa
    \floatplacement{#1}{#2}%
    \@ifundefined{fname@#1}{\floatname{#1}{#1}}{}
    \expandafter\edef\csname ftype@#1\endcsname{\value{float@type}}%
    \addtocounter{float@type}{\value{float@type}}
    \restylefloat{#1}%
    \expandafter\edef\csname fnum@#1\endcsname%
    {\expandafter\noexpand\csname fname@#1\endcsname{}
      \expandafter\noexpand\csname the#1\endcsname}
    \@ifnextchar[%]
    {\float@newx{#1}}%
    {\@ifundefined{c@#1}{\newcounter{#1}\@namedef{the#1}{\arabic{#1}}}%
      {}}%
}{%
  \scr@ifexpected{\listof}{%
    \def\listof#1#2{%
      \@ifundefined{ext@#1}{\float@error{#1}}{%
        \@namedef{l@#1}{\@dottedtocline{1}{1.5em}{2.3em}}%
        \float@listhead{#2}%
        \begingroup\setlength{\parskip}{\z@}%
        \@starttoc{\@nameuse{ext@#1}}%
        \endgroup}}%
  }{%
    \RequirePackage{tocbasic}%
    \PackageInfo{scrhack}{redefining \string\newfloat}%
    \renewcommand\newfloat[3]{%
      \ifattoclist{#3}{%
        \PackageError{scrhack}{extension `#3' already in use}{%
          Each extension may be used only once.\MessageBreak
          You, the class, or another package already uses extension
          `#3'.\MessageBreak
          \string\newfloat\space command will be ignored!}%
```

```
    }{%
      \addtotoclist[float]{#3}%
      \setuptoc{#3}{chapteratlist}%
      \@namedef{ext@#1}{#3}%
      \let\float@do=\relax
      \xdef\@tempa{\noexpand\float@exts{\the\float@exts \float@do{#3}}}%
      \@tempa
      \floatplacement{#1}{#2}%
      \@ifundefined{fname@#1}{\floatname{#1}{#1}}{}%
      \expandafter\edef\csname ftype@#1\endcsname{\value{float@type}}%
      \addtocounter{float@type}{\value{float@type}}
      \restylefloat{#1}%
      \expandafter\edef\csname fnum@#1\endcsname%
      {\expandafter\noexpand\csname fname@#1\endcsname{}
        \expandafter\noexpand\csname the#1\endcsname}%
      \@ifnextchar[%]
      {\float@newx{#1}}%
      {\@ifundefined{c@#1}{\newcounter{#1}\@namedef{the#1}{\arabic{#1}}}%
        {}}}%
  }%
  \PackageInfo{scrhack}{redefining \string\listof}%
  \renewcommand*\listof[2]{%
    \@ifundefined{ext@#1}{\float@error{#1}}{%
      \@ifundefined{l@#1}{\expandafter\let\csname l@#1\endcsname\l@figure
        \@ifundefined{l@#1}{%
          \@namedef{l@#1}{\@dottedtocline{1}{1.5em}{2.3em}}}{}%
      }{}%
      \listoftoc[{#2}]{\csname ext@#1\endcsname}%
    }%
  }%
  \scr@ifexpected{\float@addtolists}{%
    \long\def\float@addtolists#1{%
      \def\float@do##1{\addtocontents{##1}{#1}} \the\float@exts}%
  }{%
    \PackageInfo{scrhack}{undefining \string\float@addtolists}%
    \let\float@addtolists\relax
  }{%
    \PackageWarningNoLine{scrhack}{unkown \string\float@addtolists\space
      definition found!\MessageBreak
      Maybe you are using a unsupported float version}%
  }%
 }{%
  \PackageWarningNoLine{scrhack}{unknown \string\listof\space
    definition found!\MessageBreak
    Maybe you are using a unsupported float version}%
 }%
}{%
 \PackageWarningNoLine{scrhack}{unknown \string\newfloat\space
   definition found!\MessageBreak
```

```
    Maybe you are using a unsupported float version}%
}
⟨/float & body⟩
```

7.5 Der floatrow-Hack

Das floatrow-Paket verwendet das Makro `\float@listhead` zum Setzen der Überschriften. Dies wird seit KOMA-Script 3 nicht mehr empfohlen und fliegt demnächst komplett aus der Unterstützung. Stattdessen wird empfohlen, dass Pakete tocbasic unterstützen. Der Aufwand dafür ist sehr gering und wird mit vielen neuen Möglichkeiten belohnt.

Dieser Hack rüstet die tocbasic-Unterstützung für floatrow nach.

`floatrow`

```
⟨*package & option⟩
\KOMA@ifkey{floatrow}{@scrhack@floatrow}
\KOMAExecuteOptions{floatrow=true}
⟨/package & option⟩
⟨*package & body⟩
\AfterPackage*{floatrow}{%
  \KOMA@key[.scrhack.sty]{floatrow}{%
    \PackageWarning{scrhack}{option ‘floatrow’ ignored}%
    \FamilyKeyStateProcessed
  }%
  \if@scrhack@floatrow\scr@hack@load\@pkgextension{floatrow}\fi
}
⟨/package & body⟩
```

`\DeclareNewFloatType` Über die Anweisung `\DeclareNewFloatType` wird eine neue Gleitumgebung definiert. Hier muss die neue Erweiterung aus dem dritten Argument tocbasic bekannt gemacht werden.

`\listof` Über die Anweisung `\listof` wird ein Verzeichnis für Gleitumgebungen ausgegeben. Hier muss schlicht die entsprechende Anweisung von tocbasic verwendet werden.

`\float@addtolists` Diese Anweisung wird nicht länger benötigt und daher auf die ursprüngliche Definition zurückgesetzt.

```
⟨*floatrow & body⟩
\scr@ifexpected{\DeclareNewFloatType}{%
  \long\def\DeclareNewFloatType#1#2{\def\FB@captype{#1}%
    \expandafter\edef\csname ftype@#1\endcsname{\the\c@float@type}%
    \addtocounter{float@type}{\value{float@type}}%
```

```
    \@namedef{#1name}{#1}\newcounter{#1}%
    \expandafter\edef\csname fnum@#1\endcsname
    {\expandafter\noexpand\csname #1name\endcsname\nobreakspace
      \expandafter\noexpand\csname the#1\endcsname}%
    \@namedef{the#1}{\arabic{#1}}\flnew@ext{lo#1}\@namedef{fps@#1}{tbp}%
    \@namedef{l@#1}{\@dottedtocline{1}{1.5em}{2.3em}}%
    \caption@setkeys[floatrow]{newfloat}{#2}\let\FR@tmp=\relax
    \xdef\@tempa{\noexpand\flrow@types{\the\flrow@types \FR@tmp{#1}}}%
    \@tempa}%
}{%
  \scr@ifexpected{\listof}{%
    \def\listof#1#2{%
      \@ifundefined{ext@#1}{\flrow@error{Unknown float style `#1'}}{%
        \expandafter\providecommand\csname l@#1\endcsname
        {\@dottedtocline{1}{1.5em}{2.3em}}%
        \float@listhead{#2}%
        \begingroup\setlength{\parskip}{\z@}%
        \@starttoc{\@nameuse{ext@#1}}%
        \endgroup}}%
  }{%
    \RequirePackage{tocbasic}%
    \PackageInfo{scrhack}{redefining \string\DeclareNewFloatType}%
```

Eigentlich wäre es besser, wie im float-Hack einen Test vorzuschalten, ob die Dateiendung bereits in Gebrauch ist. Aber das würde voraussetzen, dass die Reihenfolge der Anweisungen geändert wird. Dazu stecke ich aber im Code von floatrow zu wenig drin. (*Note: It would be better to first test, if the new extension is already in use like done at the float hack. But I don't know the floatrow code good enough to make such a change!*)

```
    \renewcommand\DeclareNewFloatType[2]{\def\FB@captype{#1}%
      \expandafter\edef\csname ftype@#1\endcsname{\the\c@float@type}%
      \addtocounter{float@type}{\value{float@type}}%
      \@namedef{#1name}{#1}\newcounter{#1}%
      \expandafter\edef\csname fnum@#1\endcsname
      {\expandafter\noexpand\csname #1name\endcsname\nobreakspace
        \expandafter\noexpand\csname the#1\endcsname}%
      \@namedef{the#1}{\arabic{#1}}\flnew@ext{lo#1}\@namedef{fps@#1}{tbp}%
      \@namedef{l@#1}{\@dottedtocline{1}{1.5em}{2.3em}}%
      \caption@setkeys[floatrow]{newfloat}{#2}\let\FR@tmp=\relax
      \xdef\@tempa{\noexpand\flrow@types{\the\flrow@types \FR@tmp{#1}}}%
      \@tempa
      \xdef\@tempa{\noexpand\addtotoclist[float]{\@nameuse{ext@\FB@captype}}%
        \noexpand\setuptoc{\@nameuse{ext@\FB@captype}}{chapteratlist}%
      }%
      \@tempa
    }%
    \PackageInfo{scrhack}{redefining \string\listof}%
    \renewcommand*\listof[2]{%
```

```
      \@ifundefined{ext@#1}{\flrow@error{Unknown float style ‘#1’}}{%
        \@ifundefined{l@#1}{\expandafter\let\csname l@#1\endcsname\l@figure
          \@ifundefined{l@#1}{%
            \@namedef{l@#1}{\@dottedtocline{1}{1.5em}{2.3em}}}{}%
        }{}%
        \listoftoc[{#2}]{\csname ext@#1\endcsname}%
      }%
    }%
    \scr@ifexpected{\float@addtolists}{%
      \long\def\float@addtolists#1{%
        \def\float@do##1{\addtocontents{##1}{#1}} \the\float@exts}%
    }{%
      \PackageInfo{scrhack}{undefining \string\float@addtolists}%
      \let\float@addtolists\relax
    }{%
      \PackageWarningNoLine{scrhack}{unkown \string\float@addtolists\space
        definition found!\MessageBreak
        Maybe you are using a unsupported floatrow version}%
    }%
  }{%
    \PackageWarningNoLine{scrhack}{unknown \string\listof\space
      definition found!\MessageBreak
      Maybe you are using a unsupported floatrow version}%
  }%
}{%
  \PackageWarningNoLine{scrhack}{unknown \string\DeclareNewFloatType\space
    definition found!\MessageBreak
    Maybe you are using a unsupported floatrow version}%
}
⟨/floatrow & body⟩
```

7.6 Der listings-Hack

Das listings-Paket verwendet das Makro `\float@listhead` zum Setzen der Überschriften. Dies wird seit KOMA-Script 3 nicht mehr empfohlen und fliegt demnächst komplett aus der Unterstützung. Stattdessen wird empfohlen, dass Pakete tocbasic unterstützen. Der Aufwand dafür ist sehr gering und wird mit vielen neuen Möglichkeiten belohnt.

Dieser Hack rüstet die tocbasic-Unterstützung für listings nach.

`listings`

```
⟨*package & option⟩
\KOMA@ifkey{listings}{@scrhack@listings}
\KOMAExecuteOptions{listings=true}
⟨/package & option⟩
```

```
⟨*package & body⟩
\AfterPackage*{listings}{%
  \KOMA@key[.scrhack.sty]{listings}{%
    \PackageWarning{scrhack}{option 'listings' ignored}%
    \FamilyKeyStateProcessed
  }%
  \if@scrhack@listings\scr@hack@load\@pkgextension{listings}\fi
}
⟨/package & body⟩
```

`\scr@do@hack@listings`
`\lstlistoflistings`

Über dieses Macro wird das Verzeichnis der Listings gesetzt. Die gesamte Funktionalität dafür kann tocbasic überlassen werden.

`\float@addtolists`

Diese Anweisung wird nicht länger benötigt und daher auf die ursprüngliche Definition zurückgesetzt. Da listings ihre Definition mit `\AtBeginDocument` verzögert, muss dies hier ebenfalls geschehen.

```
⟨*listings & body⟩
\newcommand*{\scr@do@hack@listings}{%
  \RequirePackage{tocbasic}%
  \addtotoclist[float]{lol}%
  \setuptoc{lol}{chapteratlist}%
  \PackageInfo{scrhack}{redefining \string\lstlistoflistings}%
  \renewcommand*{\lstlistoflistings}{\listoftoc[{\lstlistlistingname}]{lol}}%
  \AtBeginDocument{%
    \scr@ifexpected{\float@addtolists}{%
      \def\float@addtolists##1{\addtocontents{lol}{##1}}%
    }{%
      \PackageInfo{scrhack}{undefining \string\float@addtolists}%
      \let\float@addtolists\relax
    }{%
      \scr@ifexpected{\float@addtolists}{%
        \def\float@addtolists##1{\addtocontents{lol}{##1}%
          \orig@float@addtolists{##1}}%
      }{%
        \PackageInfo{scrhack}{setting \string\float@addtolists\MessageBreak
          to \string\orig@float@addtolists}%
        \let\float@addtolists\orig@float@addtolists
      }{%
        \PackageWarningNoLine{scrhack}{unkown \string\float@addtolists\space
          definition found!\MessageBreak
          Maybe you are using a unsupported listings version}%
      }%
    }%
  }%
  \let\scr@do@hack@listings\relax
}
\scr@ifexpected{\lstlistoflistings}{%
```

```
  \def\lstlistoflistings{\bgroup
    \let\contentsname\lstlistlistingname
    \let\lst@temp\@starttoc \def\@starttoc##1{\lst@temp{lol}}%
    \tableofcontents \egroup}%
}{%
  \scr@do@hack@listings
}{%
  \scr@ifexpected{\lstlistoflistings}{%
    \def\lstlistoflistings{%
      \begingroup
        \@ifundefined{@restonecoltrue}{}{%
          \if@twocolumn
            \@restonecoltrue\onecolumn
          \else
            \@restonecolfalse
          \fi
        }%
        \float@listhead{\lstlistlistingname}%
        \parskip\z@\parindent\z@\parfillskip \z@ \@plus 1fil%
        \@starttoc{lol}%
        \@ifundefined{@restonecoltrue}{}{%
          \if@restonecol\twocolumn\fi
        }%
      \endgroup
    }%
  }{%
    \scr@do@hack@listings
  }{%
    \PackageWarningNoLine{scrhack}{unknown \string\lstlistoflistings\space
      definition found!\MessageBreak
      Maybe you are using a unsupported listings version}%
  }%
}
```
⟨/listings & body⟩

7.7 Der setspace-Hack

Das setspace-Paket verwendet `\@ptsize` auf ungünstige Art, indem es davon ausgeht, dass es immer eine ganze Zahl enthält. Das ist aber bei KOMA-Script keineswegs zwingend. Außerdem ist der Wert für `11pt` falsch, weil LaTeX in diesem Fall tatsächlich eine 10,95 pt-Schrift mit einem Zeilenabstand von 13,6 pt einstellt. Damit wäre der korrekte Wert für `\onehalfspacing` beispielsweise:

$$10{,}95\,\mathrm{pt}\dot{1}, 5/13{,}6\,\mathrm{pt} \equiv 1{,}208$$

Tatsächlich stellt setspace aber einen Wert von 1,213 ein, was einer effektiven Schriftgröße von 11 pt entsprechen würde. Ebenso stellt es den aktuellen

Abstand bei `\onehalfspacing` nicht relativ zur aktuellen Schriftgröße ein, sondern zur Grundschriftgröße. Damit erhält man bei

```
\documentclass[10pt]{article}
\usepackage{setspace}
\begin{document}
\large\onehalfspacing\raggedright
Fontsize: \csname f@size\endcsname pt\\
Normal baselineskip: \csname f@baselineskip\endcsname\\
baselineskip: \the\baselineskip
\end{document}
```

einen anderen Abstand als bei

```
\documentclass[11pt]{article}
\usepackage{setspace}
\begin{document}
\large\onehalfspacing\raggedright
Fontsize: \csname f@size\endcsname pt\\
Normal baselineskip: \csname f@baselineskip\endcsname\\
baselineskip: \the\baselineskip
\end{document}
```

obwohl beide Male dieselbe Schriftgröße verwendet wird. Streng genommen müsste also bei jeder Änderung der Schriftgröße der Wert Abstand angepasst werden. So weit geht dieser Hack nicht. Stattdessen wird der Wert abhängig von der tatsächlichen Schriftgröße und dem tatsächlichen Basisabstand beim Aufruf der Anweisungen eingestellt. Das ergibt immerhin in den obigen Beispielen gleiche Ergebnisse.

`setspace`

```
⟨*package & option⟩
\KOMA@ifkey{setspace}{@scrhack@setspace}
\KOMAExecuteOptions{setspace=true}
⟨/package & option⟩
⟨*package & body⟩
\AfterPackage*{setspace}{%
  \KOMA@key[.scrhack.sty]{setspace}{%
    \PackageWarning{scrhack}{option ‘setspace’ ignored}%
    \FamilyKeyStateProcessed
  }%
  \if@scrhack@setspace\scr@hack@load\@pkgextension{setspace}\fi
}
⟨/package & body⟩
```

`\onehalfspacing` Über diese Anweisung wird der eineinhalbzeilige Satz eingestellt. Ein auf drei Nachkommastellen genauer Wert erscheint mir ausreichend genau.

```
⟨*setspace & body⟩
\scr@ifexpected{\onehalfspacing}{%
  \long\def\onehalfspacing{%
    \setstretch{1.25}%  default
    \ifcase \@ptsize \relax % 10pt
      \setstretch {1.25}%
    \or % 11pt
      \setstretch {1.213}%
    \or % 12pt
      \setstretch {1.241}%
    \fi
  }%
}{%
  \renewcommand*{\onehalfspacing}{%
    \@tempdima=\dimexpr (\f@size pt)*1500/
                        (\dimexpr \f@baselineskip\relax)*\p@/1000\relax
    \expandafter\setstretch\expandafter{\strip@pt\@tempdima}%
  }%
}{%
  \PackageWarning{scrhack}{unknown \string\onehalfspacing\space
    definition found!\MessageBreak
    Maybe you are using a unsupported setpace version}%
}
```

`\doublespacing` Über diese Anweisung wird der zweizeilige Satz eingestellt. Ein auf drei Nachkommastellen genauer Wert erscheint mir ausreichend genau.

```
\scr@ifexpected{\doublespacing}{%
  \long\def\doublespacing{%
    \setstretch{1.667}%  default
    \ifcase \@ptsize \relax % 10pt
      \setstretch {1.667}%
    \or % 11pt
      \setstretch {1.618}%
    \or % 12pt
      \setstretch {1.655}%
    \fi
  }%
}{%
  \renewcommand*{\doublespacing}{%
    \@tempdima=\dimexpr (\f@size pt)*2000/
                        (\dimexpr \f@baselineskip\relax)*\p@/1000\relax
    \expandafter\setstretch\expandafter{\strip@pt\@tempdima}%
  }%
}{%
  \PackageWarning{scrhack}{unknown \string\doublespacing\space
```

```
    definition found!\MessageBreak
    Maybe you are using a unsupported setpace version}%
}
⟨/setspace & body⟩
```

7.8 Der lscape-Hack

Das lscape-Paket setzt innerhalb der `landscape`-Umgebung die Länge `\textheight` auf den Wert von `\textwidth` obwohl es auf der anderen Seite `\textwidth` nicht auf den Wert von `\textheight` setzt. Das ist inkonsequent. Da David Carlisle außerdem angibt, dass `\textwidth` nicht verändert wird, weil das zu Problemen führen konnte, ist es unverständlich, dass `\textheight` verändert wird, obwohl das ebenfalls zu Problemen führen kann, beispielsweise für showframe oder scrpage. Daher verändere ich die Definition so, dass auch `\textheight` unverändert bleibt. Dabei muss allerdings auch pdflscape berücksichtigt werden. Das ist am einfachsten mit xpatch.

`lscape`

```
⟨*package & option⟩
\RequirePackage{xpatch}%
\KOMA@ifkey{lscape}{@scrhack@lscape}%
\KOMAExecuteOptions{lscape=true}%
⟨/package & option⟩
⟨*package & body⟩
\AfterPackage*{lscape}{%
  \if@scrhack@lscape\scr@hack@load\@pkgextension{lscape}\else
    \KOMA@key[.scrhack.sty]{lscape}{%
      \PackageWarning{scrhack}{option ‘lscape’ ignored}%
      \FamilyKeyStateProcessed
    }%
  \fi
}
⟨/package & body⟩
```

`\landscape` Über diese Anweisung wird die `landscape`-Umgebung von lscape gestartet. Genau diese muss gepatcht werden. Dafür wird das Paket xpatch benötigt. Da der Patch nur geladen wird, wenn die Option dafür gesetzt ist, kann die Option daher nur ein- und ausgeschaltet werden, wenn sie bis zum Laden des Pakets aktiviert wurde.

```
⟨*lscape & body⟩
\xpatchcmd{\landscape}{\textheight=\vsize}{%
  \if@scrhack@lscape
```

Es gibt allerdings in der Tat eine Stelle, an der ein verändertet Wert von `\textheight` benötigt wird. Das ist wenn innerhalb von `\@outputpage` der Wert von `\@colht` reinitialisiert wird. Also wird das entsprechend auch noch hinein gepatcht.

```
    \scrh@LT@textheight=\vsize
    \let\scrh@LT@outputpage\@outputpage
    \def\@outputpage{\scrh@LT@outputpage\global\@colht\scrh@LT@textheight}%
  \else
    \textheight=\vsize
  \fi
}{%
  \PackageInfo{scrhack}{\string\landscape\space patched to make
    \string\textheight\space change optional}%
}{%
  \PackageWarning{scrhack}{Cannot patch \string\landscape!\MessageBreak
    Maybe you are using a unsupported lscape version}%
  \@scrhack@lscapefalse
}
```

`\scrh@LT@textheight`

```
\newlength{\scrh@LT@textheight}
⟨/lscape & body⟩
```

7.9 Optionen ausführen

Zum Schluss noch die Optionen ausführen. Im Paket wird diese Anweisung allerdings vor den Anweisungen der Hacks und den Anweisungen aus dem Abschnitt »Verwendete Anweisungen« stehen.

```
⟨*package & option⟩
\KOMAProcessOptions\relax
⟨/package & option⟩
```

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.