The **kvoptions** package

Heiko Oberdiek

<heiko.oberdiek at googlemail.com>

2010/12/23 v3.10

Abstract

This package is intended for package authors who want to use options in key value format for their package options.

Contents

1 Introduction

First I want to recommend the very good review article "A guide to key-value methods" by Joseph Wright [1]. It introduces the different key-value packages and compares them.

1.1 The beginning

This package kvoptions addresses class or package writers that want to allow their users to specify options as key value pairs, e.g.:

```
\documentclass[verbose=false,name=me]{myclass}
\usepackage[format=print]{mylayout}
```

Prominent example is package hyperref, probably the first package that offers this service. It's `\ProcessOptionsWithKV` is often copied und used in other packages, e.g. package helvet that uses this interface for its option scaled.

However copying code is not the most modern software development technique. And hyperref's code for `\ProcessOptionsWithKV` was changed to fix bugs. The version used in other packages depends on the time of copying and the awareness of hyperref's changes. Now the code is sourced out into this package and available for other package or class writers.

1.2 Overview

Package kvoptions connects package *keyval* with LaTeX's package and class *options*:

Package **keyval**	Package **kvoptions**	LaTeX kernel
\define@key	\DeclareVoidOption \DeclareStringOption \DeclareBoolOption \DeclareComplementaryOption \DisableKeyvalOption	\DeclareOption
	\DeclareDefaultOption	\DeclareOption*
	\ProcessKeyvalOptions	\ProcessOptions*
	Option patch	Class/package option system
	\SetupKeyvalOptions	

2 Usage

2.1 Process options

2.1.1 \ProcessKeyvalOptions

```
\ProcessKeyvalOptions {⟨family⟩}
\ProcessKeyvalOptions *
```

This command evaluates the global or local options of the package that are defined with keyval's interface within the family ⟨*family*⟩. It acts the same way as LaTeX's `\ProcessOptions*`. In a package unknown global options are ignored, in a class they are added to the unknown option list. The known global options and all

local options are passed to keyval's `\setkeys` command for executing the options. Unknown options are reported to the user by an error.

If the family name happens to be the same as the name of the package or class where `\ProcessKeyvalOptions` is used or the family name has previously been setup by `\SetupKeyvalOptions`, then `\ProcessKeyvalOptions` knows the family name already and you can use the star form without mandatory argument.

2.1.2 `\ProcessLocalKeyvalOptions`

```
\ProcessLocalKeyvalOptions {⟨family⟩}
\ProcessLocalKeyvalOptions *
```

This macro has the same syntax and works similar as `\ProcessKeyvalOptions`. However it ignores global options and only processes the local package options. Therefore it only can be used inside a package. An error is thrown, if it is used inside a class.

Neither of the following macros are necessary for `\ProcessKeyvalOptions`. They just help the package/class author in common tasks.

2.1.3 `\SetupKeyvalOptions`

```
\SetupKeyvalOptions {
    family = ⟨family⟩ ,
    prefix = ⟨prefix⟩
    setkeys = ⟨setkeys command⟩
}
```

This command allows to configure the default assumptions that are based on the current package or class name. LaTeX remembers this name in `\@currname`. The syntax description of the default looks a little weird, therefor an example is given for a package or class named `foobar`.

Key	Default	(example)	Used by
family	⟨`\@currname`⟩	(foobar)	`\ProcessKeyvalOptions*`
			`\DeclareBoolOption`
			`\DeclareStringOption`
prefix	⟨`\@currname`⟩@	(foobar@)	`\DeclareBoolOption`
			`\DeclareStringOption`
			`\DeclareVoidOption`
setkeys	`\setkeys`	(\kvsetkeys)	`\ProcessKeyvalOptions`
			`\ProcessLocalKeyvalOptions`

Key `setkeys` was added in version 3.9. The original `\setkeys` of package keyval is not reentrant. If an option is processed by this `\setkeys`, then the option should not call `\setkeys` again with a different family. Otherwise the next options of the first `\setkeys` call are processed with the wrong family. With key `setkeys` the macro `\kvsetkeys` can be set that does not have the problem of the original `\setkeys` of package keyval.

Probably `\setkeys` of package xkeyval is safe in this respect. But I haven't made a full analysis. At least it does not have the problem of the original `\setkeys`.

2.2 Option declarations

The options for `\ProcessKeyvalOptions` are defined by keyval's `\define@key`. Common purposes of such keys are boolean switches, they enable or disable something. Or they store a name or some kind of string in a macro. The following

commands help the user. He declares what he wants and kvoptions take care of
the key definition, resource allocation and initialization.

In order to avoid name clashes of macro names, internal commands are prefixed.
Both the prefix and the family name for the defined keys can be configured by
\SetupKeyvalOptions.

2.2.1 \DeclareStringOption

\DeclareStringOption [⟨*init*⟩] {⟨*key*⟩} [⟨*default*⟩]

A macro is created that remembers the value of the key ⟨*key*⟩. The name of the
macro consists of the option name ⟨*key*⟩ that is prefixed by the prefix (see 2.1.3).
The initial contents of the macro can be given by the first optional argument
⟨*init*⟩. The default is empty.

The the option ⟨*key*⟩ is defined. The option code just stores its value in the
macro. If the optional argument at the end of \DeclareStringOption is given,
then option ⟨*key*⟩ is defined with the default ⟨*default*⟩.

Example for a package with the following two lines:

```
\ProvidesPackage{foobar}
\DeclareStringOption[me]{name}
```

Then \DeclareStringOption defines the macro with content me, note LaTeX com-
plains if the name of the macro already exists:

```
\newcommand*{\foobar@name}{me}
```

The option definition is similar to:

```
\define@key{foobar}{name}{%
  \renewcommand*{\foobar@name}{#1}%
}
```

2.2.2 \DeclareBoolOption

\DeclareBoolOption [⟨*init*⟩] {⟨*key*⟩}

A boolean switch is generated, initialized by value ⟨*init*⟩ and the corresponding key
⟨*key*⟩ is defined. If the initialization value is not given, false is used as default.

The internal actions of \DeclareBoolOption are shown below. The example
is given for a package author who has the following two lines in his package/class:

```
\ProvidesPackage{foobar}
\DeclareBoolOption{verbose}
```

First a new switch is created:

```
\newif\iffoobar@verbose
```

and initialized:

```
\foobar@verbosefalse
```

Finally the key is defined:

```
\define@key{foobar}{verbose}[true]{...}
```

The option code configures the boolean option in the following way: If the author
specifies true or false then the switch is turned on or off respectivly. Also the
option can be given without explicit value. Then the switch is enabled. Other
values are reported as errors.

Now the switch is ready to use in the package/class, e.g.:

```
    \iffoobar@verbose
    % print verbose message
    \else
    % be quiet
    \fi
```

Users of package `\ifthen` can use the switch as boolean:

```
    \boolean{foobar@verbose}
```

2.2.3 \DeclareComplementaryOption

\DeclareComplementaryOption {⟨key⟩} {⟨parent⟩}

Sometimes contrasting names are used to characterize the two states of a boolean switch, for example draft vs. final. Both options behave like boolean options but they do not need to different switches, they should share one. `\DeclareComplementaryOption` allows this. The option ⟨key⟩ shares the switch of option ⟨parent⟩. Example:

```
    \DeclareBoolOption{draft}
    \DeclareComplementaryOption{final}{draft}
```

Then final sets the switch of draft to false, and final=false enables the draft switch.

2.2.4 \DeclareVoidOption

\DeclareVoidOption {⟨key⟩} {⟨code⟩}

`\ProcessKeyvalOptions` can be extended to recognize options that are declared in traditional way by `\DeclareOption`. But in case of the error that the user specifies a value, then this option would not recognized as key value option because of `\DeclareOption` and not detected as traditional option because of the value part. The user would get an unknown option error, difficult to understand.

`\DeclareVoidOption` solves this problem. It defines the option ⟨key⟩ as key value option. If the user specifies a value, a warning is given and the value is ignored.

The code part ⟨code⟩ is stored in a macro. The name of the macro consists of the option name ⟨key⟩ that is prefixed by the prefix (see 2.1.3). If the option is set, the macro will be executed. During the execution `\CurrentOption` is available with the current key name.

2.2.5 \DeclareDefaultOption

\DeclareDefaultOption {⟨code⟩}

This command does not define a specific key, it is the equivalent to LaTeX's `\DeclareOption*`. It allows the specification of a default action ⟨code⟩ that is invoked if an unknown option is found. While ⟨code⟩ is called, macro `\CurrentOption` contains the current option string. In addition `\CurrentOptionValue` contains the value part if the option string is parsable as key value pair, otherwise it is `\relax`. `\CurrentOptionKey` contains the key of the key value pair, or the whole option string, if it misses the equal sign.

Inside packages typical default actions are to pass unknown options to another package. Or an error message can be thrown by `\@unknownoptionerror`. This is the original error message that LaTeX gives for unkown package options. This error

message is easier to understand for the user as the error message from package keyval that is given otherwise.

A Class ignores unknown options and puts them on the unused option list. Let LaTeX do the job and just call `\OptionNotUsed`. Or the options can be passed to another class that is later loaded.

2.2.6   Local options

```
\DeclareLocalOption {⟨option⟩}
\DeclareLocalOptions {⟨option list⟩}
```

Both macros mark package options as local options. That means that they are ignored by `\ProcessKeyvalOptions` if they are given as global options. `\DeclareLocalOptions` takes one option, `\DeclareLocalOptions` expects a comma separated list of options.

2.2.7   Dynamic options

Options of LaTeX's package/class system are cleared in `\ProcessOptions`. They modify the static model of a package. For example, depending on option bookmarks package hyperref loads differently.

Options, however, defined by keyval's `\define@key` remain defined, if the options are processed by `\setkeys`. Therefore these options can also be used to model the dynamic behaviour of a package. For example, in hyperref the link colors can be changed everywhere until the end in `\end{document}`.

However package color that adds color support is necessary and it cannot be loaded after `\begin{document}`. Option colorlinks that loads color should be active until `\begin{document}` and die in some way if it is too late for loading packages. With `\DisableKeyvalOption` the package/class author can specify and configure the death of an option and controls the life period of the option.

2.2.8   `\DisableKeyvalOption`

```
\DisableKeyvalOption [⟨options⟩] {⟨family⟩} {⟨key⟩}
⟨options⟩:
   action            = undef, warning, error, or ignore    default: undef
   global or local                                         default: global
   package or class = ⟨name⟩
```

`\DisableKeyvalOption` can be called to mark the end when the option ⟨key⟩ is no longer useful. The behaviour of an option after its death can be configured by action:

undef: The option will be undefined, If it is called, `\setkeys` reports an error because of unknown key.

error or warning: Use of the option will cause an error or warning message. Also these actions require that exclusivly either the package or class name is given in options package or class.

ignore: The use of the option will silently be ignored.

The option's death can be limited to the end of the current group, if option local is given. Default is global.

The package/class author can wish the end of the option already during the package loading, then he will have static behaviour. In case of dynamic options `\DisableKeyvalOption` can be executed everywhere, also outside the package. Therefore the family name and the package/class name is usually unknown for

`\DisableKeyvalOption`. Therefore the argument for the family name is mandatory and for some actions the package/class name must be provided.

Usually a macro would configure the option death, Example:

```
\ProvidesPackage{foobar}
\DeclareBoolOption{color}
\DeclareStringOption[red]{emphcolor}
\ProcessKeyvalOptions*

\newcommand*{\foobar@DisableOption}[2]{%
  \DisableKeyvalueOption[
    action={#1},
    package=foobar
  ]{foobar}{#2}%
}

\iffoobar@color
  \RequirePackage{color}
  \renewcommand*{\emph}[1]{\textcolor{\foobar@emphcolor}{#1}}
\else
  % Option emphcolor is not wrong, if we have color support.
  % otherwise the option has no effect, but we don't want to
  % remove it. Therefore action 'ignore' is the best choice:
  \foobar@DisableOption{ignore}{emphcolor}
\fi
% No we don't need the option 'color'.
\foobar@DisableOption{warning}{color}

% With color support option 'emphcolor' will dynamically
% change the color of \emph statements.
```

2.2.9 `\AddToKeyvalOption`

`\AddToKeyvalOption` {⟨*family*⟩} {⟨*key*⟩} {⟨*code*⟩}
`\AddToKeyvalOption *` {⟨*key*⟩} {⟨*code*⟩}

The code for an existing key ⟨*key*⟩ of family ⟨*family*⟩ is extended by code ⟨*code*⟩. In the starred form the current family setting is used, see `\ProcessKeyvalOptions*`.

2.3   Global vs. local options

Options that are given for `\documentclass` are called global options. They are known to the class and all packages. A package may make use of a global option and marks it as used. The advantage for the user is the freedom to specify options both in the `\documentclass` or `\usepackage` commands.

However global options are shared with the class options and options of all other packages. Thus there can be the same option with different semantics for different packages and classes. As example, package **bookmark** knows option **open** that specifies whether the bookmarks are opened or closed initially. It's values are **true** or **false**. Since KOMA-Script version 3.00 the KOMA classes also introduces option **open** with values **right** and **any** and a complete different meaning.

Such conflicts can be resolved by marking all or part of options as local by `\DeclareLocalOption` or `\DeclareLocalOptions`. Then the packages ignores global occurences of these options. Package **kvoptions** provides two methods:

- `\ProcessLocalKeyvalOptions` automatically uses all options as local options. It ignores all global options.

- `\DeclareLocalOption` or `\DeclareLocalOptions` marks options as local options. `\ProcessKeyvalOptions` will then ignore global occurences for these local options.

Since version 1.5 package bookmark uses the latter method. It checks global and local option places for driver options and limits all other options as local options. Thus the class option open of KOMA-Script is not misread as option for package bookmark.

2.4 Summary of internal macros

The `\Declare...Option` commands define macros, additionally to the macros generated by the key definition. These macros can be used by the package/class author. The name of the macros starts with the prefix ⟨*prefix*⟩ that can be configured by `\SetupKeyvalOptions`.

Declare ⟨*key*⟩	Defined macro	Description
`\DeclareStringOption`	`\`⟨*prefix*⟩⟨*key*⟩	holds the string
`\DeclareBoolOption`	`\if`⟨*prefix*⟩⟨*key*⟩	boolean switch
	`\`⟨*prefix*⟩⟨*key*⟩`false`	disable switch
	`\`⟨*prefix*⟩⟨*key*⟩`true`	enable switch
`\DeclareComplementaryOption`	`\`⟨*prefix*⟩⟨*key*⟩`false`	enable parent switch
	`\`⟨*prefix*⟩⟨*key*⟩`true`	disable parent switch
`\DeclareVoidOption`	`\`⟨*prefix*⟩⟨*key*⟩	holds the action

2.5 plain TeX

Package keyval is also usable in plain TeX with the help of file `miniltx.tex`. Some features of this package kvoptions might also be useful for plain TeX. If LaTeX is not found, `\ProcessKeyvalOptions` and option patch are disabled. Before using the option declaration commands `\Declare...Option`, `\SetupKeyvalOptions` must be used.

3 Example

The following example defined a package that serves some private color management. A boolean option print enables print mode without colors. An option emph redefines `\emph` to print in the given color. And the driver can be specified by option driver.

```
 1 ⟨*example⟩
 2     % Package identification
 3     % ----------------------
 4 \NeedsTeXFormat{LaTeX2e}
 5 \ProvidesPackage{example-mycolorsetup}[2010/12/23 Managing my colors]
 6
 7 \RequirePackage{ifpdf}
 8 \RequirePackage{kvoptions}
 9
10     % Option declarations
11     % -------------------
12
13 \SetupKeyvalOptions{
14   family=MCS,
15   prefix=MCS@
16 }
17     % Use a shorter family name and prefix
18
19     % Option print
20 \DeclareBoolOption{print}
21     % is the same as
22     % \DeclareBoolOption[false]{print}
23
```

```
24     % Option driver
25 \ifpdf
26   \DeclareStringOption[pdftex]{driver}
27 \else
28   \DeclareStringOption[dvips]{driver}
29 \fi
30
31     % Alternative interface for driver options
32 \DeclareVoidOption{dvips}{\SetupDriver}
33 \DeclareVoidOption{dvipdfm}{\SetupDriver}
34 \DeclareVoidOption{pdftex}{\SetupDriver}
35     % In \SetupDriver we take the current option \CurrentOption
36     % and pass it to the driver option.
37     % The \expandafter commands expand \CurrentOption at the
38     % time, when \SetupDriver is executed and \CurrentOption
39     % has the correct meaning.
40 \newcommand*{\SetupDriver}{%
41   \expandafter\@SetupDriver\expandafter{\CurrentOption}%
42 }
43 \newcommand*{\@SetupDriver}[1]{%
44   \setkeys{MCS}{driver={#1}}%
45 }
46
47     % Option emph
48     % An empty value means, we want to have no color for \emph.
49     % If the user specifies option emph without value, the red is used.
50 \DeclareStringOption{emph}[red]
51     % is the same as
52     % \DeclareStringOption[]{emph}[red]
53
54     % Default option rule
55 \DeclareDefaultOption{%
56   \ifx\CurrentOptionValue\relax
57     \PackageWarningNoLine{\@currname}{%
58       Unknown option '\CurrentOption'\MessageBreak
59       is passed to package 'color'%
60     }%
61     % Pass the option to package color.
62     % Again it is better to expand \CurrentOption.
63     \expandafter\PassOptionsToPackage
64     \expandafter{\CurrentOption}{color}%
65   \else
66     % Package color does not take options with values.
67     % We provide the standard LaTeX error.
68     \@unknownoptionerror
69   \fi
70 }
71
72     % Process options
73     % ---------------
74 \ProcessKeyvalOptions*
75
76     % Implementation depending on option values
77     % -----------------------------------------
78     % Code for print mode
79 \ifMCS@print
80   \PassOptionsToPackage{monochrome}{color}
81     % tells package color to use black and white
82 \fi
83
84 \RequirePackage[\MCS@driver]{color}
85     % load package color with the correct driver
```

```
86
87    % \emph setup
88 \ifx\MCS@emph\@empty
89    % \@empty is a predefined macro with empty contents.
90    % the option value of option emph is empty, thus
91    % we do not want a redefinition of \emph.
92 \else
93  \renewcommand*{\emph}[1]{%
94    \textcolor{\MCS@emph}{#1}%
95  }
96 \fi
97 ⟨/example⟩
```

4   Package options

The package kvoptions knows two package options patch and debugshow. The op-
tions of package kvoptions are intended for authors, not for package/class writers.
Inside a package it is too late for option patch and debugshow enables some mes-
sages that are perhaps useful for the debugging phase. Also LATEX is unhappy if
a package is loaded later again with options that are previously not given. Thus
package and class authors, stay with \RequirePackage{kvoptions} without op-
tions.

Option patch loads package kvoptions-patch.

4.1   Package kvoptions-patch

LATEX's system of package/class options has some severe limitations that especially
affects the value part if options are used as pair of key and value.

- Spaces are removed, regardless where:

    ```
    \documentclass[box=0 0 400 600]{article}
    ```

  Now each package will see box=00400600 as global option.

- In the previous case also braces would not help:

    ```
    \documentclass[box={0 0 400 600}]{article}
    ```

  The result is an error message:

    ```
    ! LaTeX Error: Missing \begin{document}.
    ```

  As local option, however, it works if the package knows about key value
  options (By using this package, for example).

- The requirements on robustness are extremly high. LATEX expands the op-
  tion. All that will not work as environment name will break also as option.
  Even a \relax will generate an error message:

    ```
    ! Missing \endcsname inserted.
    ```

  Of course, LATEX does not use its protecting mechanisms. On contrary
  \protect itself will cause errors.

- The options are expanded. But perhaps the package will do that, because
  it has to setup some things before? Example hyperref:

    ```
    \usepackage[pdfauthor=M\"uller]{hyperref}
    ```

  Package hyperref does not see M\"uller but its expansion and it does not
  like it, you get many warnings

```
Token not allowed in a PDFDocEncoded string
```

And the title becomes: `Mu127uller`. Therefore such options must usually be given after package hyperref is loaded:

```
\usepackage{hyperref}
\hypersetup{pdfauthor=Fran\c coise M\"uller}
```

As package option it will even break with `Fran\c coise` because of the cedilla `\c c`, it is not robust enough.

For users that do not want with this limitations the package offers package kvoptions-patch. It patches LaTeX's option system and tries to teach it also to handle options that are given as pairs of key and value and to prevent expansion. It can already be used at the very beginning, before `\documentclass`:

```
\RequirePackage{kvoptions-patch}
\documentclass[pdfauthor=Fran\c coise M\"uller]{article}
\usepackage{hyperref}
```

The latest time is before the package where you want to use problematic values:

```
\usepackage{kvoptions-patch}
\usepackage[Fran\c coise M\"uller]{hyperref}
```

Some remarks:

- The patch requires ε-TeX, its `\unexpanded` feature is much to nice. It is possible to work around using token registers. But the code becomes longer, slower, more difficult to read and maintain. The package without option patch works and will work without ε-TeX.

- The code for the patch is quite long, there are many test cases. Thus the probability for bugs is probably not too small.

- Since 2008/10/18 v3.0 package kvoptions-patch is available. Before option patch of package kvoptions must be used instead. I think, the solution as standalone package kvoptions-patch is cleaner and avoids option clashes.

4.2 Option **debugshow**

The name of this option follows the convention of packages multicol, tabularx, and tracefnt. Currently it prints the setting of boolean options, declared by `\DeclareBoolOption` in the `.log` file, if that boolean option is used. You can activate the option by

- `\PassOptionsToPackage{debugshow}{kvoptions}`
  Put this somewhere before package kvoptions is loaded first, e.g. before `\documentclass`.

- `\RequirePackage[debugshow]{kvoptions}`
  Before `\documentclass` even an author has to use `\RequirePackage`. `\usepackage` only works after `\documentclass`.

The preferred method is `\PassOptionsToPackage`, because it does not force the package loading and does not disturb, if the package is not loaded later at all.

5 Limitations

5.1 Compatibility

5.1.1 Package **kvoptions-patch** vs. package **xkvltxp**

Package xkvltxp from the xkeyval project has the same goal as package kvoptions-patch and to patch LaTeX's kernel commands in order to get better support for key value options. Of course they cannot be used both. The user must decide, which method he prefers. Package kvoptions-patch aborts itself, if it detects that xkvltxp is already loaded.

However package xkvltxp and kvoptions can be used together, example:

```
\usepackage{xkvltxp}
\usepackage[...]{foobar} % foobar using kvoptions
```

The other way should work, too.

Package kvoptions-patch tries to catch more situations and to be more robust. For example, during the comparison of options it normalizes them by removing spaces around = and the value. Thus the following is not reported as option clash:

```
\RequirePackage{kvoptions-patch}
\documentclass{article}

\usepackage[scaled=0.7]{helvet}
\usepackage[scaled = 0.7]{helvet}

\begin{document}
\end{document}
```

5.2 Limitations

5.2.1 Option comparisons

In some situations LaTeX compares option lists, e.g. option clash check, \@ifpackagewith, or \@ifclasswith. Apart from catcode and sanitizing problems of option patch, there is another problem. LaTeX does not know about the type and default values of options in key value style. Thus an option clash is reported, even if the key value has the same meaning:

```
\usepackage[scaled]{helvet} % default is .95
\usepackage[.95]{helvet}
\usepackage[0.95]{helvet}
```

5.2.2 Option list parsing with package **kvoptions-patch**

With package kvoptions-patch the range of possible values in key value specifications is much large, for example the comma can be used, if enclosed in curly braces.

Other packages, especially the packages that uses their own process option code can be surprised to find tokens inside options that they do not expect and errors would be the consequence. To avoid errors the options, especially the unused option list is sanitized. That means the list will only contain tokens with catcode 12 (other) and perhaps spaces (catcode 10). This allows a safe parsing for other packages. But a comma in the value part is no longer protected by curly braces because they have lost their special meaning. This is the price for compatibility.

Example:

```
\RequirePackage{kvoptions-patch}
\documentclass[a={a,b,c},b]{article}
\begin{document}
\end{document}
```

Result:

```
LaTeX Warning: Unused global option(s):
    [a={a,c},b].
```

6 Implementation

6.1 Preamble

98 ⟨*package⟩

Reload check and identification. Reload check, especially if the package is not used with LATEX.

```
 99 \begingroup\catcode61\catcode48\catcode32=10\relax%
100   \catcode13=5 % ^^M
101   \endlinechar=13 %
102   \catcode35=6 % #
103   \catcode39=12 % '
104   \catcode44=12 % ,
105   \catcode45=12 % -
106   \catcode46=12 % .
107   \catcode58=12 % :
108   \catcode64=11 % @
109   \catcode123=1 % {
110   \catcode125=2 % }
111   \expandafter\let\expandafter\x\csname ver@kvoptions.sty\endcsname
112   \ifx\x\relax % plain-TeX, first loading
113   \else
114     \def\empty{}%
115     \ifx\x\empty % LaTeX, first loading,
116       % variable is initialized, but \ProvidesPackage not yet seen
117     \else
118       \expandafter\ifx\csname PackageInfo\endcsname\relax
119         \def\x#1#2{%
120           \immediate\write-1{Package #1 Info: #2.}%
121         }%
122       \else
123         \def\x#1#2{\PackageInfo{#1}{#2, stopped}}%
124       \fi
125       \x{kvoptions}{The package is already loaded}%
126       \aftergroup\endinput
127     \fi
128   \fi
129 \endgroup%
```

Package identification:

```
130 \begingroup\catcode61\catcode48\catcode32=10\relax%
131   \catcode13=5 % ^^M
132   \endlinechar=13 %
133   \catcode35=6 % #
134   \catcode39=12 % '
135   \catcode40=12 % (
136   \catcode41=12 % )
137   \catcode44=12 % ,
138   \catcode45=12 % -
139   \catcode46=12 % .
140   \catcode47=12 % /
141   \catcode58=12 % :
142   \catcode64=11 % @
143   \catcode91=12 % [
144   \catcode93=12 % ]
145   \catcode123=1 % {
146   \catcode125=2 % }
```

```
147  \expandafter\ifx\csname ProvidesPackage\endcsname\relax
148    \def\x#1#2#3[#4]{\endgroup
149      \immediate\write-1{Package: #3 #4}%
150      \xdef#1{#4}%
151    }%
152  \else
153    \def\x#1#2[#3]{\endgroup
154      #2[{#3}]%
155      \ifx#1\@undefined
156        \xdef#1{#3}%
157      \fi
158      \ifx#1\relax
159        \xdef#1{#3}%
160      \fi
161    }%
162  \fi
163 \expandafter\x\csname ver@kvoptions.sty\endcsname
164 \ProvidesPackage{kvoptions}%
165    [2010/12/23 v3.10 Keyval support for LaTeX options (HO)]%
```

Catcodes

```
166 \begingroup\catcode61\catcode48\catcode32=10\relax%
167   \catcode13=5 % ^^M
168   \endlinechar=13 %
169   \catcode123=1 % {
170   \catcode125=2 % }
171   \catcode64=11 % @
172   \def\x{\endgroup
173     \expandafter\edef\csname KVO@AtEnd\endcsname{%
174       \endlinechar=\the\endlinechar\relax
175       \catcode13=\the\catcode13\relax
176       \catcode32=\the\catcode32\relax
177       \catcode35=\the\catcode35\relax
178       \catcode61=\the\catcode61\relax
179       \catcode64=\the\catcode64\relax
180       \catcode123=\the\catcode123\relax
181       \catcode125=\the\catcode125\relax
182     }%
183   }%
184 \x\catcode61\catcode48\catcode32=10\relax%
185 \catcode13=5 % ^^M
186 \endlinechar=13 %
187 \catcode35=6 % #
188 \catcode64=11 % @
189 \catcode123=1 % {
190 \catcode125=2 % }
191 \def\TMP@EnsureCode#1#2{%
192   \edef\KVO@AtEnd{%
193     \KVO@AtEnd
194     \catcode#1=\the\catcode#1\relax
195   }%
196   \catcode#1=#2\relax
197 }
198 \TMP@EnsureCode{1}{14}% ^^A (comment)
199 \TMP@EnsureCode{2}{14}% ^^A (comment)
200 \TMP@EnsureCode{33}{12}% !
201 \TMP@EnsureCode{39}{12}% '
202 \TMP@EnsureCode{40}{12}% (
203 \TMP@EnsureCode{41}{12}% )
204 \TMP@EnsureCode{42}{12}% *
205 \TMP@EnsureCode{44}{12}% ,
206 \TMP@EnsureCode{45}{12}% -
```

```
207 \TMP@EnsureCode{46}{12}% .
208 \TMP@EnsureCode{47}{12}% /
209 \TMP@EnsureCode{58}{12}% :
210 \TMP@EnsureCode{62}{12}% >
211 \TMP@EnsureCode{91}{12}% [
212 \TMP@EnsureCode{93}{12}% ]
213 \TMP@EnsureCode{94}{7}% ^ (superscript)
214 \TMP@EnsureCode{96}{12}% `
215 \edef\KVO@AtEnd{\KVO@AtEnd\noexpand\endinput}
```

External resources. The package extends the support for key value pairs of package \keyval to package options. Thus the package needs to be loaded anyway. and we use it for \SetupKeyvalOptions. AFAIK this does not disturb users of xkeyval.

```
216 \@ifundefined{define@key}{%
217   \RequirePackage{keyval}\relax
218 }{}
```

Macro \DeclareLocalOptions parses a comma separated key list and uses \comma@parse of package kvsetkeys, version 1.3.

```
219 \RequirePackage{ltxcmds}[2010/12/02]
220 \RequirePackage{kvsetkeys}[2007/09/29]
```

Provide macros for plain TeX.

```
221 \@ifundefined{@x@protect}{%
222   \def\@x@protect#1\fi#2#3{%
223     \fi\protect#1%
224   }%
225   \let\@typeset@protect\relax
226 }{}
227 \@ifundefined{@currname}{%
228   \def\@currname{}%
229 }{}
230 \@ifundefined{@currext}{%
231   \def\@currext{}%
232 }{}
```

Options Option debugshow enables additional lines of code that prints information into the .log file.

```
233 \DeclareOption{debugshow}{\catcode\@ne=9 }
```

```
234 \DeclareOption{patch}{%
235   \AtEndOfPackage{%
236     \RequirePackage{kvoptions-patch}[2010/12/23]%
237   }%
238 }
```

Optionen auswerten:

```
239 \ProcessOptions\relax
```

6.2   Option declaration macros

6.2.1   \SetupKeyvalOptions

The family for the key value pairs can be setup once and is remembered later. The package name seems a reasonable default for the family key, if it is not set by the package author.

\KVO@family We cannot store the family setting in one macro, because the package should be usable for many other packages, too. Thus we remember the family setting in a

macro, whose name contains the package name with extension, a key in LaTeX's class/package system.

```
240 \define@key{KVO}{family}{%
241   \expandafter\edef\csname KVO@family@%
242       \@currname.\@currext\endcsname{#1}%
243 }
244 \def\KVO@family{%
245   \@ifundefined{KVO@family@\@currname.\@currext}{%
246     \@currname
247   }{%
248     \csname KVO@family@\@currname.\@currext\endcsname
249   }%
250 }
```

\KVO@prefix   The value settings of options that are declared by \DeclareBoolOption and \DeclareStringOption need to be saved in macros. in the first case this is a switch \if⟨prefix⟩⟨key⟩, in the latter case a macro \⟨prefix⟩⟨key⟩. The prefix can be configured, by prefix that is declared here. The default is the package name with @ appended.

```
251 \define@key{KVO}{prefix}{%
252   \expandafter\edef\csname KVO@prefix@%
253       \@currname.\@currext\endcsname{#1}%
254 }
255 \def\KVO@prefix{%
256   \ltx@ifundefined{KVO@prefix@\@currname.\@currext}{%
257     \@currname @%
258   }{%
259     \csname KVO@prefix@\@currname.\@currext\endcsname
260   }%
261 }

262 \define@key{KVO}{setkeys}{%
263   \expandafter\def\csname KVO@setkeys@%
264       \@currname.\@currext\endcsname{#1}%
265 }
```

\KVO@setkeys

```
266 \def\KVO@setkeys{%
267   \ltx@IfUndefined{KVO@setkeys@\@currname.\@currext}{%
268     \setkeys
269   }{%
270     \csname KVO@setkeys@\@currname.\@currext\endcsname
271   }%
272 }
```

\SetupKeyvalOptions   The argument of \SetupKeyvalOptions expects a key value list, known keys are family and prefix.

```
273 \newcommand*{\SetupKeyvalOptions}{%
274   \kvsetkeys{KVO}%
275 }
```

6.2.2   \DeclareBoolOption

\DeclareBoolOption   Usually options of boolean type can be given by the user without value and this means a setting to *true*. We follow this convention here. Also it simplifies the user interface.

The switch is created and initialized with *false*. The default setting can be overwritten by the optional argument.

LaTeX's \newif does not check for already defined macros, therefore we add this check here to prevent the user from accidently redefining of TeX's primitives and other macros.

```
276 \newcommand*{\DeclareBoolOption}[2][false]{%
277   \KVO@ifdefinable{if\KVO@prefix#2}{%
278     \KVO@ifdefinable{\KVO@prefix#2true}{%
279       \KVO@ifdefinable{\KVO@prefix#2false}{%
280         \csname newif\expandafter\endcsname
281         \csname if\KVO@prefix#2\endcsname
282         \@ifundefined{\KVO@prefix#2#1}{%
283           \PackageWarning{kvoptions}{%
284             Initialization of option '#2' failed,\MessageBreak
285             cannot set boolean option to '#1',\MessageBreak
286             use 'true' or 'false', now using 'false'%
287           }%
288         }{%
289           \csname\KVO@prefix#2#1\endcsname
290         }%
291         \begingroup
292           \edef\x{\endgroup
293             \noexpand\define@key{\KVO@family}{#2}[true]{%
294               \noexpand\KVO@boolkey{\@currname}%
295               \ifx\@currext\@clsextension
296                 \noexpand\@clsextension
297               \else
298                 \noexpand\@pkgextension
299               \fi
300               {\KVO@prefix}{#2}{####1}%
301             }%
302           }%
303           \x
304         }%
305       }%
306     }%
307 }
```

The first argument is the key name, the second the key that must be a boolean option with the same current family and prefix. A new switch is not created for the new key, we have already a switch. Instead we define switch setting commands to work on the parent switch.

```
308 \newcommand*{\DeclareComplementaryOption}[2]{%
309   \@ifundefined{if\KVO@prefix#2}{%
310     \PackageError{kvoptions}{%
311       Cannot generate option code for '#1',\MessageBreak
312       parent switch '#2' does not exist%
313     }{%
314       You are inside %
315       \ifx\@currext\@clsextension class\else package\fi\space
316       '\@currname.\@currext'.\MessageBreak
317       '\KVO@family' is used as familiy %
318       for the keyval options.\MessageBreak
319       '\KVO@prefix' serves as prefix %
320       for internal switch macros.\MessageBreak
321       \MessageBreak
322       \@ehc
323     }%
324   }{%
325     \KVO@ifdefinable{\KVO@prefix#1true}{%
326       \KVO@ifdefinable{\KVO@prefix#1false}{%
327         \expandafter\let\csname\KVO@prefix#1false\expandafter\endcsname
328           \csname\KVO@prefix#2true\endcsname
329         \expandafter\let\csname\KVO@prefix#1true\expandafter\endcsname
330           \csname\KVO@prefix#2false\endcsname
```

The same code part as in \DeclareBoolOption can now be used.

```
331         \begingroup
```

18

```
332          \edef\x{\endgroup
333            \noexpand\define@key{\KVO@family}{#1}[true]{%
334              \noexpand\KVO@boolkey{\@currname}%
335              \ifx\@currext\@clsextension
336                \noexpand\@clsextension
337              \else
338                \noexpand\@pkgextension
339              \fi
340              {\KVO@prefix}{#1}{####1}%
341            }%
342          }%
343        \x
344      }%
345    }%
346  }%
347 }
```

\KVO@ifdefinable   Generate the command token LaTeX's `\@ifdefinable` expects.

```
348 \def\KVO@ifdefinable#1{%
349   \expandafter\@ifdefinable\csname #1\endcsname
350 }
```

\KVO@boolkey   We check explicitly for `true` and `false` to prevent the user from accidently calling other macros.

#1	package/class name
#2	\@pkgextension/\@clsextension
#3	prefix
#4	key name
#5	new value

```
351 \def\KVO@boolkey#1#2#3#4#5{%
352   \edef\KVO@param{#5}%
353   \ltx@onelevel@sanitize\KVO@param
354   \ifx\KVO@param\KVO@true
355     \expandafter\@firstofone
356   \else
357     \ifx\KVO@param\KVO@false
358       \expandafter\expandafter\expandafter\@firstofone
359     \else
360       \ifx#2\@clsextension
361         \expandafter\ClassWarning
362       \else
363         \expandafter\PackageWarning
364       \fi
365       {#1}{%
366         Value '\KVO@param' is not supported by\MessageBreak
367         option '#4'%
368       }%
369       \expandafter\expandafter\expandafter\@gobble
370     \fi
371   \fi
372   {%
373     ^^A\ifx#2\@clsextension
374     ^^A  \expandafter\ClassInfo
375     ^^A\else
376     ^^A  \expandafter\PackageInfo
377     ^^A\fi
378     ^^A{#1}{[option] #4=\KVO@param}%
379     \csname#3#4\KVO@param\endcsname
380   }%
381 }
```

19

The macros `\KVO@true` and `\KVO@false` are used for string comparisons. After `\ltx@onelevel@sanitize` we have only tokens with catcode 12 (other).

```
382 \def\KVO@true{true}
383 \def\KVO@false{false}
384 \ltx@onelevel@sanitize\KVO@true
385 \ltx@onelevel@sanitize\KVO@false
```

6.2.3 \DeclareStringOption

```
386 \newcommand*{\DeclareStringOption}[2][]{%
387   \@ifnextchar[{%
388     \KVO@DeclareStringOption{#1}{#2}@%
389   }{%
390     \KVO@DeclareStringOption{#1}{#2}{}[]%
391   }%
392 }
```

```
393 \def\KVO@DeclareStringOption#1#2#3[#4]{%
394   \KVO@ifdefinable{\KVO@prefix#2}{%
395     \@namedef{\KVO@prefix#2}{#1}%
396     \begingroup
397       \ifx\\#3\\%
398         \toks@{}%
399       \else
400         \toks@{[{#4}]}%
401       \fi
402       \edef\x{\endgroup
403         \noexpand\define@key{\KVO@family}{#2}\the\toks@{%
404           ^^A\begingroup
405           ^^A  \toks@{####1}%
406           ^^A  \ifx\@currext\@clsextension
407           ^^A    \noexpand\ClassInfo
408           ^^A  \else
409           ^^A    \noexpand\PackageInfo
410           ^^A  \fi
411           ^^A  {\@currname}{%
412           ^^A    [option] #2={\noexpand\the\toks@}%
413           ^^A  }%
414           ^^A\endgroup
415           \noexpand\def
416           \expandafter\noexpand\csname\KVO@prefix#2\endcsname{####1}%
417         }%
418       }%
419     \x
420   }%
421 }
```

6.2.4 \DeclareVoidOption

```
422 \newcommand*{\DeclareVoidOption}[2]{%
423   \begingroup
424     \let\next\@gobbletwo
425     \KVO@ifdefinable{\KVO@prefix#1}{%
426       \let\next\@firstofone
427     }%
428   \expandafter\endgroup
429   \next{%
430     \begingroup
```

```
431     \edef\x{\endgroup
432       \noexpand\define@key{\KVO@family}{#1}[\KVO@VOID@]{%
433         \noexpand\KVO@voidkey{\@currname}%
434         \ifx\@currext\@clsextension
435           \noexpand\@clsextension
436         \else
437           \noexpand\@pkgextension
438         \fi
439         {#1}%
440         {####1}%
441         \expandafter\noexpand\csname\KVO@prefix#1\endcsname
442       }%
443     }%
444   \x
445   \begingroup
446     \toks@{#2}%
447   \expandafter\endgroup
448   \expandafter\def
449   \csname\KVO@prefix#1\expandafter\endcsname
450   \expandafter{\the\toks@}%
451   }%
452 }
453 \def\KVO@VOID@{@VOID@}
```

	#1	package/class name
	#2	\@pkgextension/\@clsextension
\KVO@voidkey	#3	key name
	#4	default (@VOID@)
	#5	macro with option code

```
454 \def\KVO@voidkey#1#2#3#4{%
455   \def\CurrentOption{#3}%
456   \begingroup
457     \def\x{#4}%
458   \expandafter\endgroup
459   \ifx\x\KVO@VOID@
460   \else
461     \ifx#2\@clsextension
462       \expandafter\ClassWarning
463     \else
464       \expandafter\PackageWarning
465     \fi
466     {#1}{%
467       Unexpected value for option '#3'\MessageBreak
468       is ignored%
469     }%
470   \fi
471 ^^A\ifx#2\@clsextension
472 ^^A  \expandafter\ClassInfo
473 ^^A\else
474 ^^A  \expandafter\PackageInfo
475 ^^A\fi
476 ^^A{#1}{[option] #3}%
477 }
```

6.2.5 \DeclareDefaultOption

\DeclareDefaultOption

```
478 \newcommand*{\DeclareDefaultOption}{%
479   \@namedef{KVO@default@\@currname.\@currext}%
480 }
```

6.2.6 \DeclareLocalOptions

\DeclareLocalOptions

```
481 \newcommand*{\DeclareLocalOptions}[1]{%
482   \comma@parse{#1}\KVO@DeclareLocalOption
483 }
```

\KVO@DeclareLocalOption

```
484 \def\KVO@DeclareLocalOption#1{%
485   \expandafter\def\csname KVO@local@\KVO@family @#1\endcsname{}%
486 }
```

6.3 Dynamic options

6.3.1 \DisableKeyvalOption

```
487 \SetupKeyvalOptions{%
488   family=KVOdyn,%
489   prefix=KVOdyn@%
490 }
491 \DeclareBoolOption[true]{global}
492 \DeclareComplementaryOption{local}{global}
493 \DeclareStringOption[undef]{action}
494 \let\KVOdyn@name\relax
495 \let\KVOdyn@ext\@empty
496 \define@key{KVOdyn}{class}{%
497   \def\KVOdyn@name{#1}%
498   \let\KVOdyn@ext\@clsextension
499 }
500 \define@key{KVOdyn}{package}{%
501   \def\KVOdyn@name{#1}%
502   \let\KVOdyn@ext\@pkgextension
503 }
504 \newcommand*{\DisableKeyvalOption}[3][]{%
505   \begingroup
506     \kvsetkeys{KVOdyn}{#1}%
507     \def\x{\endgroup}%
508     \@ifundefined{KVO@action@\KVOdyn@action}{%
509       \PackageError{kvoptions}{%
510         Unknown disable action %
511         `\expandafter\strip@prefix\meaning\KVOdyn@action'\MessageBreak
512         for option `#3' in keyval family `#2'%
513       }\@ehc
514     }{%
515       \csname KVO@action@\KVOdyn@action\endcsname{#2}{#3}%
516     }%
517   \x
518 }
519 \def\KVO@action@undef#1#2{%
520   \edef\x{\endgroup
521     \ifKVOdyn@global\global\fi
522     \let
523     \expandafter\noexpand\csname KV@#1@#2\endcsname
524     \relax
525     \ifKVOdyn@global\global\fi
526     \let
527     \expandafter\noexpand\csname KV@#1@#2@default\endcsname
528     \relax
529   }%
530   ^^A\PackageInfo{kvoptions}{%
531   ^^A  [option] key `#2' of family `#1'\MessageBreak
532   ^^A  is disabled (undef, \ifKVOdyn@global global\else local\fi)%
533   ^^A}%
```

```
534 }
535 \def\KVO@action@ignore#1#2{%
536   \edef\x{\endgroup
537     \ifKVOdyn@global\global\fi
538     \let
539     \expandafter\noexpand\csname KV@#1@#2\endcsname
540     \noexpand\@gobble
541     \ifKVOdyn@global\global\fi
542     \let
543     \expandafter\noexpand\csname KV@#1@#2@default\endcsname
544     \noexpand\@empty
545   }%
546   ^^A\PackageInfo{kvoptions}{%
547   ^^A  [option] key '#2' of family '#1'\MessageBreak
548   ^^A  is disabled (ignore, \ifKVOdyn@global global\else local\fi)%
549   ^^A}%
550 }
551 \def\KVO@action@error{%
552   \KVO@do@action{error}%
553 }
554 \def\KVO@action@warning{%
555   \KVO@do@action{warning}%
556 }
```

#1   error or warning
#2   ⟨family⟩
#3   ⟨key⟩

```
557 \def\KVO@do@action#1#2#3{%
558   \ifx\KVOdyn@name\relax
559     \PackageError{kvoptions}{%
560       Action type '#1' needs package/class name\MessageBreak
561       for key '#3' in family '#2'%
562     }\@ehc
563   \else
564     \edef\x{\endgroup
565       \noexpand\define@key{#2}{#3}[]{%
566         \expandafter\noexpand\csname KVO@disable@#1\endcsname
567         {\KVOdyn@name}\noexpand\KVOdyn@ext{#3}%
568       }%
569       \ifKVOdyn@global
570         \global\let
571         \expandafter\noexpand\csname KV@#2@#3\endcsname
572         \expandafter\noexpand\csname KV@#2@#3\endcsname
573         \global\let
574         \expandafter\noexpand\csname KV@#2@#3@default\endcsname
575         \expandafter\noexpand\csname KV@#2@#3@default\endcsname
576       \fi
577     }%
578     ^^A\ifx\KVOdyn@ext\@clsextension
579     ^^A  \expandafter\ClassInfo
580     ^^A\else
581     ^^A   \expandafter\PackageInfo
582     ^^A\fi
583     ^^A{\KVOdyn@name}{%
584     ^^A  [option] key '#3' of family '#2'\MessageBreak
585     ^^A  is disabled (#1, \ifKVOdyn@global global\else local\fi)%
586     ^^A}%
587   \fi
588 }
589 \def\KVO@disable@error#1#2#3{%
590   \ifx#2\@clsextension
591     \expandafter\ClassError
592   \else
```

```
593        \expandafter\PackageError
594      \fi
595      {#1}{%
596        Option '#3' is given too late,\MessageBreak
597        now the option is ignored%
598      }\@ehc
599  }
600  \def\KVO@disable@warning#1#2#3{%
601      \ifx#2\@clsextension
602        \expandafter\ClassWarning
603      \else
604        \expandafter\PackageWarning
605      \fi
606      {#1}{%
607        Option '#3' is already consumed\MessageBreak
608        and has no effect%
609      }%
610  }
```

6.4   Change option code

6.4.1   \AddToKeyvalOption

\AddToKeyvalOption

```
611  \newcommand*{\AddToKeyvalOption}{%
612      \@ifstar{%
613        \begingroup
614          \edef\x{\endgroup
615            \noexpand\KVO@AddToKeyvalOption{\KVO@family}%
616          }%
617        \x
618      }%
619      \KVO@AddToKeyvalOption
620  }
```

\KVO@AddToKeyvalOption

```
621  \def\KVO@AddToKeyvalOption#1#2{%
622      \@ifundefined{KV@#1@#2}{%
623        \PackageWarning{kvoptions}{%
624          Key '#2' of family '#1' does not exist.\MessageBreak
625          Ignoring \string\AddToKeyvalOption
626        }%
627        \@gobble
628      }{%
629        \edef\KVO@next{%
630          \noexpand\KVO@@AddToKeyvalOption
631          \expandafter\noexpand\csname KV@#1@#2\endcsname
632        }%
633        \afterassignment\KVO@next
634        \def\KVO@temp##1%
635      }%
636  }
```

\KVO@@AddToKeyvalOption

```
637  \def\KVO@@AddToKeyvalOption#1{%
638      \begingroup
639        \toks@\expandafter{#1{##1}}%
640        \toks@\expandafter{\the\expandafter\toks@\KVO@temp{##1}}%
641        \edef\x{\endgroup
642          \noexpand\def\noexpand#1####1{\the\toks@}%
643        }%
644      \x
645  }
```

6.5 Process options

6.5.1 \ProcessKeyvalOptions

\ProcessKeyvalOptions If the optional star is given, we get the family name and expand it for safety.

```
646 \newcommand*{\ProcessKeyvalOptions}{%
647   \@ifstar{%
648     \begingroup
649       \edef\x{\endgroup
650         \noexpand\KVO@ProcessKeyvalOptions{\KVO@family}%
651       }%
652     \x
653   }%
654   \KVO@ProcessKeyvalOptions
655 }

656 \def\KVO@ProcessKeyvalOptions#1{%
657   \let\@tempc\relax
658   \let\KVO@temp\@empty
```

Add any global options that are known to KV to the start of the list being built in \KVO@temp and mark them used (by removing them from the unused option list).

```
659   \ifx\@currext\@clsextension
660   \else
661     \ifx\@classoptionslist\relax
662     \else
663       \@for\KVO@CurrentOption:=\@classoptionslist\do{%
664         \@ifundefined{KV@#1@\expandafter\KVO@getkey
665                 \KVO@CurrentOption=\@nil}{%
666       }{%
667         \@ifundefined{KVO@local@#1@\expandafter\KVO@getkey
668                 \KVO@CurrentOption=\@nil}{%
669           \ifx\KVO@Patch Y%
670             \edef\KVO@temp{%
671               \etex@unexpanded\expandafter{%
672                 \KVO@temp
673               }%
674               ,%
675               \etex@unexpanded\expandafter{%
676                 \KVO@CurrentOption
677               }%
678               ,%
679             }%
680             \ltx@onelevel@sanitize\KVO@CurrentOption
681           \else
682             \edef\KVO@temp{%
683               \KVO@temp
684               ,%
685               \KVO@CurrentOption
686               ,%
687             }%
688           \fi
689           \@expandtwoargs\@removeelement\KVO@CurrentOption
690             \@unusedoptionlist\@unusedoptionlist
691         }{}%
692       }%
693     }%
694   \fi
695   \fi
```

Now stick the package options at the end of the list and wrap in a call to \setkeys. A class ignores unknown global options, we must remove them to prevent error messages from \setkeys.

```
696    \begingroup
697      \toks\tw@{}%
698      \@ifundefined{opt@\@currname.\@currext}{%
699        \toks@\expandafter{\KVO@temp}%
700      }{%
701        \toks@\expandafter\expandafter\expandafter{%
702          \csname opt@\@currname.\@currext\endcsname
703        }%
704        \ifx\@currext\@clsextension
705          \edef\CurrentOption{\the\toks@}%
706          \toks@\expandafter{\KVO@temp}%
707          \@for\CurrentOption:=\CurrentOption\do{%
708            \@ifundefined{%
709              KV@#1@\expandafter\KVO@getkey\CurrentOption=\@nil
710            }{%
```

A class puts not used options in the unused option list unless there is a default handler.

```
711              \@ifundefined{KVO@default@\@currname.\@currext}{%
712                \ifx\KVO@Patch Y%
713                  \ltx@onelevel@sanitize\CurrentOption
714                \fi
715                \ifx\@unusedoptionlist\@empty
716                  \global\let\@unusedoptionlist\CurrentOption
717                \else
718                  \expandafter\expandafter\expandafter\gdef
719                  \expandafter\expandafter\expandafter\@unusedoptionlist
720                  \expandafter\expandafter\expandafter{%
721                    \expandafter\@unusedoptionlist
722                    \expandafter,\CurrentOption
723                  }%
724                \fi
725              }{%
726                \toks\tw@\expandafter{%
727                  \the\toks\expandafter\tw@\expandafter,\CurrentOption
728                }%
729              }%
730            }{%
731              \toks@\expandafter{%
732                \the\expandafter\toks@\expandafter,\CurrentOption
733              }%
734            }%
735          }%
736        \else
```

Without default action we pass all options to \setkeys. Otherwise we have to check which options are known. These are passed to \setkeys. For the others the default action is performed.

```
737          \@ifundefined{KVO@default@\@currname.\@currext}{%
738            \toks@\expandafter\expandafter\expandafter{%
739              \expandafter\KVO@temp\the\toks@
740            }%
741          }{%
742            \edef\CurrentOption{\the\toks@}%
743            \toks@\expandafter{\KVO@temp}%
744            \@for\CurrentOption:=\CurrentOption\do{%
745              \@ifundefined{%
746                KV@#1@\expandafter\KVO@getkey\CurrentOption=\@nil
747              }{%
748                \toks\tw@\expandafter{%
749                  \the\toks\expandafter\tw@\expandafter,\CurrentOption
750                }%
751              }{%
```

```
752          \toks@\expandafter{%
753            \the\expandafter\toks@\expandafter,\CurrentOption
754          }%
755        }%
756      }%
757    }%
758    \fi
759  }%
760  \edef\KVO@temp{\endgroup
761    \noexpand\KVO@calldefault{\the\toks\tw@}%
762    \noexpand\KVO@setkeys{#1}{\the\toks@}%
763  }%
764  \KVO@temp
```

Some cleanup of `\ProcessOptions`.

```
765    \let\CurrentOption\@empty
766    \AtEndOfPackage{\let\@unprocessedoptions\relax}%
767 }
```

6.5.2 \ProcessLocalKeyvalOptions

`\ProcessLocalKeyvalOptions` If the optional star is given, we get the family name and expand it for safety.

```
768 \newcommand*{\ProcessLocalKeyvalOptions}{%
769   \@ifstar{%
770     \begingroup
771       \edef\x{\endgroup
772         \noexpand\KVO@ProcessLocalKeyvalOptions{\KVO@family}%
773       }%
774     \x
775   }%
776   \KVO@ProcessLocalKeyvalOptions
777 }
```

```
778 \def\KVO@ProcessLocalKeyvalOptions#1{%
779   \let\@tempc\relax
780   \let\KVO@temp\@empty
```

Check if `\ProcessLocalKeyvalOptions` is called inside a package.

```
781   \ifx\@currext\@pkgextension
782   \else
783     \PackageError{kvoptions}{%
784       \string\ProcessLocalKeyvalOptions\space is intended for packages only%
785     }\@ehc
786   \fi
```

The package options are put into toks register `\toks@`.

```
787   \begingroup
788     \toks\tw@{}%
789     \@ifundefined{opt@\@currname.\@currext}{%
790       \toks@\expandafter{\KVO@temp}%
791     }{%
792       \toks@\expandafter\expandafter\expandafter{%
793         \csname opt@\@currname.\@currext\endcsname
794       }%
```

Without default action we pass all options to `\setkeys`. Otherwise we have to check which options are known. These are passed to `\setkeys`. For the others the default action is performed.

```
795       \@ifundefined{KVO@default@\@currname.\@currext}{%
796         \toks@\expandafter\expandafter\expandafter{%
797           \expandafter\KVO@temp\the\toks@
798         }%
799       }{%
800         \edef\CurrentOption{\the\toks@}%
```

```
801        \toks@\expandafter{\KVO@temp}%
802        \@for\CurrentOption:=\CurrentOption\do{%
803          \@ifundefined{%
804            KV@#1@\expandafter\KVO@getkey\CurrentOption=\@nil
805          }{%
806            \toks\tw@\expandafter{%
807              \the\toks\expandafter\tw@\expandafter,\CurrentOption
808            }%
809          }{%
810            \toks@\expandafter{%
811              \the\expandafter\toks@\expandafter,\CurrentOption
812            }%
813          }%
814        }%
815      }%
816    }%
817    \edef\KVO@temp{\endgroup
818      \noexpand\KVO@calldefault{\the\toks\tw@}%
819      \noexpand\KVO@setkeys{#1}{\the\toks@}%
820    }%
821  \KVO@temp
```

Some cleanup of \ProcessOptions.

```
822    \let\CurrentOption\@empty
823    \AtEndOfPackage{\let\@unprocessedoptions\relax}%
824 }
```

6.5.3 Helper macros

\KVO@getkey Extract the key part of a key=value pair.

```
825 \def\KVO@getkey#1=#2\@nil{#1}
```

\KVO@calldefault

```
826 \def\KVO@calldefault#1{%
827   \begingroup
828     \def\x{#1}%
829   \expandafter\endgroup
830   \ifx\x\@empty
831   \else
832     \@for\CurrentOption:=#1\do{%
833       \ifx\CurrentOption\@empty
834       \else
835         \expandafter\KVO@setcurrents\CurrentOption=\@nil
836         \@nameuse{KVO@default@\@currname.\@currext}%
837       \fi
838     }%
839   \fi
840 }
```

\KVO@setcurrents Extract the key part of a key=value pair.

```
841 \def\KVO@setcurrents#1=#2\@nil{%
842   \def\CurrentOptionValue{#2}%
843   \ifx\CurrentOptionValue\@empty
844     \let\CurrentOptionKey\CurrentOption
845     \let\CurrentOptionValue\relax
846   \else
847     \edef\CurrentOptionKey{\zap@space#1 \@empty}%
848     \expandafter\KVO@setcurrentvalue\CurrentOption\@nil
849   \fi
850 }
```

\KV@setcurrentvalue   Here the value part is parsed. Package keyval's `\KV@@sp@def` helps in removing spaces at the begin and end of the value.

```
851 \def\KVO@setcurrentvalue#1=#2\@nil{%
852   \KV@@sp@def\CurrentOptionValue{#2}%
853 }
```

6.6   plain TeX

Disable LaTeX stuff.

```
854 \begingroup\expandafter\expandafter\expandafter\endgroup
855 \expandafter\ifx\csname documentclass\endcsname\relax
856   \def\ProcessKeyvalOptions{%
857     \@ifstar{}\@gobble
858   }%
859 \fi
860 \KVO@AtEnd%
861 ⟨/package⟩
```

6.7   Package **kvoptions-patch**

```
862 ⟨*patch⟩
863 \NeedsTeXFormat{LaTeX2e}
864 \begingroup\catcode61\catcode48\catcode32=10\relax%
865   \catcode13=5 % ^^M
866   \endlinechar=13 %
867   \catcode123=1 % {
868   \catcode125=2 % }
869   \catcode64=11 % @
870   \def\x{\endgroup
871     \expandafter\edef\csname KVO@AtEnd\endcsname{%
872       \endlinechar=\the\endlinechar\relax
873       \catcode13=\the\catcode13\relax
874       \catcode32=\the\catcode32\relax
875       \catcode35=\the\catcode35\relax
876       \catcode61=\the\catcode61\relax
877       \catcode64=\the\catcode64\relax
878       \catcode123=\the\catcode123\relax
879       \catcode125=\the\catcode125\relax
880     }%
881   }%
882 \x\catcode61\catcode48\catcode32=10\relax%
883 \catcode13=5 % ^^M
884 \endlinechar=13 %
885 \catcode35=6 % #
886 \catcode64=11 % @
887 \catcode123=1 % {
888 \catcode125=2 % }
889 \def\TMP@EnsureCode#1#2{%
890   \edef\KVO@AtEnd{%
891     \KVO@AtEnd
892     \catcode#1=\the\catcode#1\relax
893   }%
894   \catcode#1=#2\relax
895 }
896 \TMP@EnsureCode{39}{12}% '
897 \TMP@EnsureCode{40}{12}% (
898 \TMP@EnsureCode{41}{12}% )
899 \TMP@EnsureCode{43}{12}% +
900 \TMP@EnsureCode{44}{12}% ,
901 \TMP@EnsureCode{45}{12}% -
902 \TMP@EnsureCode{46}{12}% .
```

```
903 \TMP@EnsureCode{47}{12}% /
904 \TMP@EnsureCode{58}{12}% :
905 \TMP@EnsureCode{60}{12}% <
906 \TMP@EnsureCode{62}{12}% >
907 \TMP@EnsureCode{91}{12}% [
908 \TMP@EnsureCode{93}{12}% ]
909 \TMP@EnsureCode{96}{12}% `
910 \TMP@EnsureCode{124}{12}% |
911 \edef\KVO@AtEnd{\KVO@AtEnd\noexpand\endinput}
912 \ProvidesPackage{kvoptions-patch}%
913   [2010/12/23 v3.10 LaTeX patch for keyval options (HO)]%
```

Check for ε-TeX.

```
914 \begingroup\expandafter\expandafter\expandafter\endgroup
915 \expandafter\ifx\csname eTeXversion\endcsname\relax
916   \PackageWarningNoLine{kvoptions-patch}{%
917     Package loading is aborted, because e-TeX is missing%
918   }%
919   \expandafter\KVO@AtEnd
920 \fi%
```

Package etexcmds for \etex@unexpanded.

```
921 \RequirePackage{etexcmds}[2007/09/09]
922 \ifetex@unexpanded
923 \else
924   \PackageError{kvoptions-patch}{%
925     Could not find eTeX's \string\unexpanded.\MessageBreak
926     Try adding \string\RequirePackage\string{etexcmds\string} %
927     before \string\documentclass%
928   }\@ehd
929   \expandafter\KVO@AtEnd
930 \fi%
```

Check for package xkvltxp.

```
931 \@ifpackageloaded{xkvltxp}{%
932   \PackageWarningNoLine{kvoptions}{%
933     Option `patch' cannot be used together with\MessageBreak
934     package `xkvltxp' that is already loaded.\MessageBreak
935     Therefore package loading is aborted%
936   }%
937   \KVO@AtEnd
938 }{}%
939 \def\@if@ptions#1#2#3{%
940   \begingroup
941     \KVO@normalize\KVO@temp{#3}%
942     \edef\x{\endgroup
943       \noexpand\@if@pti@ns{%
944         \detokenize\expandafter\expandafter\expandafter{%
945           \csname opt@#2.#1\endcsname
946         }%
947       }{%
948         \detokenize\expandafter{\KVO@temp}%
949       }%
950     }%
951   \x
952 }
953 \def\@pass@ptions#1#2#3{%
954   \KVO@normalize\KVO@temp{#2}%
955   \@ifundefined{opt@#3.#1}{%
956     \expandafter\gdef\csname opt@#3.#1%
957         \expandafter\endcsname\expandafter{%
958       \KVO@temp
959     }%
960   }{%
```

```
961    \expandafter\gdef\csname opt@#3.#1%
962            \expandafter\expandafter\expandafter\endcsname
963            \expandafter\expandafter\expandafter{%
964        \csname opt@#3.#1\expandafter\endcsname\expandafter,\KVO@temp
965     }%
966   }%
967 }
968 \def\ProcessOptions{%
969   \let\ds@\@empty
970   \@ifundefined{opt@\@currname.\@currext}{%
971     \let\@curroptions\@empty
972   }{%
973     \expandafter\expandafter\expandafter\def
974     \expandafter\expandafter\expandafter\@curroptions
975     \expandafter\expandafter\expandafter{%
976       \csname opt@\@currname.\@currext\endcsname
977     }%
978   }%
979   \@ifstar\KVO@xprocess@ptions\KVO@process@ptions
980 }
981 \def\KVO@process@ptions{%
982   \@for\CurrentOption:=\@declaredoptions\do{%
983     \ifx\CurrentOption\@empty
984     \else
985       \begingroup
986         \ifx\@currext\@clsextension
987           \toks@{}%
988         \else
989           \toks@\expandafter{\@classoptionslist,}%
990         \fi
991         \toks\tw@\expandafter{\@curroptions}%
992         \edef\x{\endgroup
993           \noexpand\in@{,\CurrentOption,}{,\the\toks@\the\toks\tw@,}%
994         }%
995       \x
996       \ifin@
997         \KVO@use@ption
998         \expandafter\let\csname ds@\CurrentOption\endcsname\@empty
999       \fi
1000    \fi
1001   }%
1002   \KVO@process@pti@ns
1003 }
1004 \def\KVO@xprocess@ptions{%
1005   \ifx\@currext\@clsextension
1006   \else
1007     \@for\CurrentOption:=\@classoptionslist\do{%
1008       \ifx\CurrentOption\@empty
1009       \else
1010         \KVO@in@\CurrentOption\@declaredoptions
1011         \ifin@
1012           \KVO@use@ption
1013           \expandafter\let\csname ds@\CurrentOption\endcsname\@empty
1014         \fi
1015       \fi
1016     }%
1017   \fi
1018   \KVO@process@pti@ns
1019 }
1020 \def\KVO@in@#1#2{%
1021   \in@false
```

```
1022    \begingroup
1023      \@for\x:=#2\do{%
1024        \ifx\x#1\relax
1025          \in@true
1026        \fi
1027      }%
1028      \edef\x{\endgroup
1029        \ifin@
1030          \noexpand\in@true
1031        \fi
1032      }%
1033    \x
1034 }

1035 \def\KVO@process@pti@ns{%
1036    \@for\CurrentOption:=\@curroptions\do{%
1037      \@ifundefined{ds@\KVO@SanitizedCurrentOption}{%
1038        \KVO@use@ption
1039        \default@ds
1040      }%
1041      \KVO@use@ption
1042    }%
1043    \@for\CurrentOption:=\@declaredoptions\do{%
1044      \expandafter\let\csname ds@\CurrentOption\endcsname\relax
1045    }%
1046    \let\CurrentOption\@empty
1047    \let\@fileswith@pti@ns\@@fileswith@pti@ns
1048    \AtEndOfPackage{\let\@unprocessedoptions\relax}%
1049 }

1050 \def\KVO@use@ption{%
1051    \begingroup
1052      \edef\x{\endgroup
1053        \noexpand\@removeelement{%
1054          \detokenize\expandafter{\CurrentOption}%
1055        }{%
1056          \detokenize\expandafter{\@unusedoptionlist}%
1057        }%
1058      }%
1059    \x\@unusedoptionlist
1060    \csname ds@\KVO@SanitizedCurrentOption\endcsname
1061 }

1062 \def\OptionNotUsed{%
1063    \ifx\@currext\@clsextension
1064      \xdef\@unusedoptionlist{%
1065        \ifx\@unusedoptionlist\@empty
1066        \else
1067          \detokenize\expandafter{\@unusedoptionlist,}%
1068        \fi
1069        \detokenize\expandafter{\CurrentOption}%
1070      }%
1071    \fi
1072 }
```

Variant of `\ExecuteOptions` that better protects `\CurrentOption`.

```
1073 \def\CurrentOption@SaveLevel{0}
1074 \def\ExecuteOptions{%
1075    \expandafter\KVO@ExecuteOptions
1076        \csname CurrentOption@\CurrentOption@SaveLevel\endcsname
1077 }
1078 \def\KVO@ExecuteOptions#1#2{%
1079    \let#1\CurrentOption
1080    \edef\CurrentOption@SaveLevel{%
1081      \the\numexpr\CurrentOption@SaveLevel+1%
```

32

```
1082    }%
1083    \@for\CurrentOption:=#2\do{%
1084      \csname ds@\CurrentOption\endcsname
1085    }%
1086    \edef\CurrentOption@SaveLevel{%
1087      \the\numexpr\CurrentOption@SaveLevel-1%
1088    }%
1089    \let\CurrentOption#1%
1090 }
1091 \def\KVO@fileswith@pti@ns#1[#2]#3[#4]{%
1092    \ifx#1\@clsextension
1093      \ifx\@classoptionslist\relax
1094        \KVO@normalize\KVO@temp{#2}%
1095        \expandafter\gdef\expandafter\@classoptionslist\expandafter{%
1096          \KVO@temp
1097        }%
1098        \def\reserved@a{%
1099          \KVO@onefilewithoptions{#3}[{#2}][{#4}]#1%
1100          \@documentclasshook
1101        }%
1102      \else
1103        \def\reserved@a{%
1104          \KVO@onefilewithoptions{#3}[{#2}][{#4}]#1%
1105        }%
1106      \fi
1107    \else
1108      \begingroup
1109        \let\KVO@temp\relax
1110        \let\KVO@onefilewithoptions\relax
1111        \let\@pkgextension\relax
1112        \def\reserved@b##1,{%
1113          \ifx\@nil##1\relax
1114          \else
1115            \ifx\relax##1\relax
1116            \else
1117              \KVO@onefilewithoptions{##1}[{\KVO@temp}][{#4}]%
1118              \@pkgextension
1119            \fi
1120            \expandafter\reserved@b
1121          \fi
1122        }%
1123        \edef\reserved@a{\zap@space#3 \@empty}%
1124        \edef\reserved@a{\expandafter\reserved@b\reserved@a,\@nil,}%
1125        \toks@{#2}%
1126        \def\KVO@temp{\the\toks@}%
1127      \edef\reserved@a{\endgroup \reserved@a}%
1128    \fi
1129    \reserved@a
1130 }
1131 \def\KVO@onefilewithoptions#1[#2][#3]#4{%
1132    \@pushfilename
1133    \xdef\@currname{#1}%
1134    \global\let\@currext#4%
1135    \expandafter\let\csname\@currname.\@currext-h@@k\endcsname\@empty
1136    \let\CurrentOption\@empty
1137    \@reset@ptions
1138    \makeatletter
1139    \def\reserved@a{%
1140      \@ifl@aded\@currext{#1}{%
1141        \@if@ptions\@currext{#1}{#2}{%
1142        }{%
1143          \begingroup
```

33

```
1144        \@ifundefined{opt@#1.\@currext}{%
1145          \def\x{}%
1146        }{%
1147          \edef\x{%
1148            \expandafter\expandafter\expandafter\strip@prefix
1149            \expandafter\meaning\csname opt@#1.\@currext\endcsname
1150          }%
1151        }%
1152        \def\y{#2}%
1153        \edef\y{\expandafter\strip@prefix\meaning\y}%
1154        \@latex@error{Option clash for \@cls@pkg\space #1}{%
1155          The package #1 has already been loaded %
1156          with options:\MessageBreak
1157          \space\space[\x]\MessageBreak
1158          There has now been an attempt to load it %
1159           with options\MessageBreak
1160          \space\space[\y]\MessageBreak
1161          Adding the global options:\MessageBreak
1162          \space\space
1163                \x,\y\MessageBreak
1164          to your \noexpand\documentclass declaration may fix this.%
1165          \MessageBreak
1166          Try typing \space <return> \space to proceed.%
1167        }%
1168        \endgroup
1169      }%
1170    }{%
1171      \@pass@ptions\@currext{#2}{#1}%
1172      \global\expandafter
1173      \let\csname ver@\@currname.\@currext\endcsname\@empty
1174      \InputIfFileExists
1175        {\@currname.\@currext}%
1176        {}%
1177        {\@missingfileerror\@currname\@currext}%
1178      \let\@unprocessedoptions\@@unprocessedoptions
1179      \csname\@currname.\@currext-h@@k\endcsname
1180      \expandafter\let\csname\@currname.\@currext-h@@k\endcsname
1181            \@undefined
1182      \@unprocessedoptions
1183    }%
1184    \@ifl@ter\@currext{#1}{#3}{%
1185    }{%
1186      \@latex@warning@no@line{%
1187        You have requested,\on@line, %
1188        version\MessageBreak
1189          #3' of \@cls@pkg\space #1,\MessageBreak
1190        but only version\MessageBreak
1191         `\csname ver@#1.\@currext\endcsname'\MessageBreak
1192        is available%
1193      }%
1194    }%
1195    \ifx\@currext\@clsextension\let\LoadClass\@twoloadclasserror\fi
1196    \@popfilename
1197    \@reset@ptions
1198  }%
1199  \reserved@a
1200 }

1201 \def\@unknownoptionerror{%
1202  \@latex@error{%
1203    Unknown option `\KVO@SanitizedCurrentOption' %
1204    for \@cls@pkg\space`\@currname'%
1205  }{%
```

```
1206     The option `\KVO@SanitizedCurrentOption' was not declared in %
1207     \@cls@pkg\space`\@currname', perhaps you\MessageBreak
1208     misspelled its name. %
1209     Try typing \space <return> %
1210     \space to proceed.%
1211   }%
1212 }

1213 \def\@@unprocessedoptions{%
1214   \ifx\@currext\@pkgextension
1215     \@ifundefined{opt@\@currname.\@currext}{%
1216       \let\@curroptions\@empty
1217     }{%
1218       \expandafter\let\expandafter\@curroptions
1219           \csname opt@\@currname.\@currext\endcsname
1220     }%
1221     \@for\CurrentOption:=\@curroptions\do{%
1222         \ifx\CurrentOption\@empty\else\@unknownoptionerror\fi
1223     }%
1224   \fi
1225 }

1226 \def\KVO@SanitizedCurrentOption{%
1227   \expandafter\strip@prefix\meaning\CurrentOption
1228 }
     Normalize option list.
1229 \def\KVO@normalize#1#2{%
1230   \let\KVO@result\@empty
1231   \KVO@splitcomma#2,\@nil
1232   \let#1\KVO@result
1233 }
1234 \def\KVO@splitcomma#1,#2\@nil{%
1235   \KVO@ifempty{#1}{}{%
1236     \KVO@checkkv#1=\@nil
1237   }%
1238   \KVO@ifempty{#2}{}{\KVO@splitcomma#2\@nil}%
1239 }
1240 \def\KVO@ifempty#1{%
1241   \expandafter\ifx\expandafter\\\detokenize{#1}\\%
1242     \expandafter\@firstoftwo
1243   \else
1244     \expandafter\@secondoftwo
1245   \fi
1246 }
1247 \def\KVO@checkkv#1=#2\@nil{%
1248   \KVO@ifempty{#2}{%
1249     % option without value
1250     \edef\KVO@x{\zap@space#1 \@empty}%
1251     \ifx\KVO@x\@empty
1252       % ignore empty option
1253     \else
1254       % append to list
1255       \edef\KVO@result{%
1256         \etex@unexpanded\expandafter{\KVO@result},\KVO@x
1257       }%
1258     \fi
1259   }{%
1260     % #1: "key", #2: "value="
1261     % add key part
1262     \edef\KVO@result{%
1263       \etex@unexpanded\expandafter{\KVO@result},%
1264       \zap@space#1 \@empty
1265     }%
```

35

```
1266      \futurelet\@let@token\KVO@checkfirsttok#2 \@nil| = \@nil|\KVO@nil
1267    }%
1268 }
1269 \def\KVO@checkfirsttok{%
1270    \ifx\@let@token\bgroup
1271      % no space at start
1272      \expandafter\KVO@removelastspace\expandafter=%
1273      % "<value><spaceopt>= \@nil"
1274    \else
1275      \expandafter\KVO@checkfirstA
1276    \fi
1277 }
1278 \def\KVO@checkfirstA#1 #2\@nil{%
1279    \KVO@ifempty{#2}{%
1280      \KVO@removelastspace=#1 \@nil
1281    }{%
1282      \KVO@ifempty{#1}{%
1283        \KVO@removelastspace=#2\@nil
1284      }{%
1285        \KVO@removelastspace=#1 #2\@nil
1286      }%
1287    }%
1288 }
1289 \def\KVO@removelastspace#1 = \@nil|#2\KVO@nil{%
1290    \KVO@ifempty{#2}{%
1291      \edef\KVO@result{%
1292        \etex@unexpanded\expandafter{\KVO@result}%
1293        \etex@unexpanded\expandafter{\KVO@removegarbage#1\KVO@nil}%
1294      }%
1295    }{%
1296      \edef\KVO@result{%
1297        \etex@unexpanded\expandafter{\KVO@result}%
1298        \etex@unexpanded{#1}%
1299      }%
1300    }%
1301 }
1302 \def\KVO@removegarbage#1= \@nil#2\KVO@nil{#1}%
```
Arguments #1 and #2 are macros.
```
1303 \def\KVO@removeelement#1#2{%
1304    \begingroup
1305      \toks@={}%
1306      \@for\x:=#2\do{%
1307        \ifx\x\@empty
1308        \else
1309          \ifx\x#1\relax
1310          \else
1311            \edef\t{\the\toks@}%
1312            \ifx\t\@empty
1313            \else
1314              \toks@\expandafter{\the\toks@,}%
1315            \fi
1316            \toks@\expandafter{\the\expandafter\toks@\x}%
1317          \fi
1318        \fi
1319      }%
1320      \edef\x{\endgroup
1321        \def\noexpand#2{\the\toks@}%
1322      }%
1323    \x
1324 }
1325 \let\@@fileswith@pti@ns\KVO@fileswith@pti@ns
1326 \ifx\@fileswith@pti@ns\@badrequireerror
```

```
1327 \else
1328   \let\@fileswith@pti@ns\KVO@fileswith@pti@ns
1329 \fi
```

\KVO@Patch

```
1330 \let\KVO@Patch=Y

1331 \KVO@AtEnd%
1332 ⟨/patch⟩
```

7 Test

7.1 Preface for standard catcode check

```
1333 ⟨*test1⟩
1334 \input miniltx.tex\relax
1335 ⟨/test1⟩
```

7.2 Catcode checks for loading

```
1336 ⟨*test1⟩
1337 \catcode`\{=1 %
1338 \catcode`\}=2 %
1339 \catcode`\#=6 %
1340 \catcode`\@=11 %
1341 \expandafter\ifx\csname count@\endcsname\relax
1342   \countdef\count@=255 %
1343 \fi
1344 \expandafter\ifx\csname @gobble\endcsname\relax
1345   \long\def\@gobble#1{}%
1346 \fi
1347 \expandafter\ifx\csname @firstofone\endcsname\relax
1348   \long\def\@firstofone#1{#1}%
1349 \fi
1350 \expandafter\ifx\csname loop\endcsname\relax
1351   \expandafter\@firstofone
1352 \else
1353   \expandafter\@gobble
1354 \fi
1355 {%
1356   \def\loop#1\repeat{%
1357     \def\body{#1}%
1358     \iterate
1359   }%
1360   \def\iterate{%
1361     \body
1362       \let\next\iterate
1363     \else
1364       \let\next\relax
1365     \fi
1366     \next
1367   }%
1368   \let\repeat=\fi
1369 }%
1370 \def\RestoreCatcodes{}
1371 \count@=0 %
1372 \loop
1373   \edef\RestoreCatcodes{%
1374     \RestoreCatcodes
1375     \catcode\the\count@=\the\catcode\count@\relax
1376   }%
1377 \ifnum\count@<255 %
1378   \advance\count@ 1 %
```

```
1379 \repeat
1380
1381 \def\RangeCatcodeInvalid#1#2{%
1382   \count@=#1\relax
1383   \loop
1384     \catcode\count@=15 %
1385   \ifnum\count@<#2\relax
1386     \advance\count@ 1 %
1387   \repeat
1388 }
1389 \def\RangeCatcodeCheck#1#2#3{%
1390   \count@=#1\relax
1391   \loop
1392     \ifnum#3=\catcode\count@
1393     \else
1394       \errmessage{%
1395         Character \the\count@\space
1396         with wrong catcode \the\catcode\count@\space
1397         instead of \number#3%
1398       }%
1399     \fi
1400   \ifnum\count@<#2\relax
1401     \advance\count@ 1 %
1402   \repeat
1403 }
1404 \def\space{ }
1405 \expandafter\ifx\csname LoadCommand\endcsname\relax
1406   \def\LoadCommand{\input kvoptions.sty\relax}%
1407 \fi
1408 \def\Test{%
1409   \RangeCatcodeInvalid{0}{47}%
1410   \RangeCatcodeInvalid{58}{64}%
1411   \RangeCatcodeInvalid{91}{96}%
1412   \RangeCatcodeInvalid{123}{255}%
1413   \catcode`\@=12 %
1414   \catcode`\\=0 %
1415   \catcode`\%=14 %
1416   \LoadCommand
1417   \RangeCatcodeCheck{0}{36}{15}%
1418   \RangeCatcodeCheck{37}{37}{14}%
1419   \RangeCatcodeCheck{38}{47}{15}%
1420   \RangeCatcodeCheck{48}{57}{12}%
1421   \RangeCatcodeCheck{58}{63}{15}%
1422   \RangeCatcodeCheck{64}{64}{12}%
1423   \RangeCatcodeCheck{65}{90}{11}%
1424   \RangeCatcodeCheck{91}{91}{15}%
1425   \RangeCatcodeCheck{92}{92}{0}%
1426   \RangeCatcodeCheck{93}{96}{15}%
1427   \RangeCatcodeCheck{97}{122}{11}%
1428   \RangeCatcodeCheck{123}{255}{15}%
1429   \RestoreCatcodes
1430 }
1431 \Test
1432 \csname @@end\endcsname
1433 \end
1434 ⟨/test1⟩
1435 ⟨*test2⟩
1436 \NeedsTeXFormat{LaTeX2e}
1437 \makeatletter
1438 \catcode`\@=11 %
1439 \def\RestoreCatcodes{}
1440 \count@=0 %
```

```
1441 \loop
1442   \edef\RestoreCatcodes{%
1443     \RestoreCatcodes
1444     \catcode\the\count@=\the\catcode\count@\relax
1445   }%
1446 \ifnum\count@<255 %
1447   \advance\count@\@ne
1448 \repeat
1449
1450 \def\RangeCatcodeInvalid#1#2{%
1451   \count@=#1\relax
1452   \loop
1453     \catcode\count@=15 %
1454   \ifnum\count@<#2\relax
1455     \advance\count@\@ne
1456   \repeat
1457 }
1458 \def\Test#1{%
1459   \RangeCatcodeInvalid{0}{47}%
1460   \RangeCatcodeInvalid{58}{64}%
1461   \RangeCatcodeInvalid{91}{96}%
1462   \RangeCatcodeInvalid{123}{255}%
1463   \catcode`\@=12 %
1464   \catcode`\\=0 %
1465   \catcode`\{=1 %
1466   \catcode`\}=2 %
1467   \catcode`\#=6 %
1468   \catcode`\[=12 %
1469   \catcode`\]=12 %
1470   \catcode`\%=14 %
1471   \catcode`\ =10 %
1472   \catcode13=5 %
1473   #1\relax
1474   \RestoreCatcodes
1475 }
1476 \Test{\RequirePackage{kvoptions-patch}}%
1477 \Test{\RequirePackage{kvoptions}}%
1478 \csname @@end\endcsname
1479 ⟨/test2⟩
1480 ⟨*test3⟩
1481 \NeedsTeXFormat{LaTeX2e}
1482 \makeatletter
1483 \RequirePackage{kvoptions}[2010/12/23]
1484 \def\msg#{\immediate\write16}
1485 \define@key{testfamily}{testkey}{%
1486   \msg{[testfamily/testkey/#1]}%
1487 }
1488 \define@key{testfamily}{testdefaultkey}[testdefault]{%
1489   \msg{[testfamily/testdefaultkey/#1]}%
1490 }
1491 \AddToKeyvalOption{testfamily}{testkey}{%
1492   \msg{[addition/#1]}%
1493 }
1494 \AddToKeyvalOption{testfamily}{testdefaultkey}{%
1495   \msg{[addition/#1]}%
1496 }
1497 \setkeys{testfamily}{%
1498   testkey=testA,%
1499   testdefaultkey=testB,%
1500   testdefaultkey,%
1501 }
1502 \SetupKeyvalOptions{%
```

```
1503   family=testfamily%
1504 }
1505 \AddToKeyvalOption*{testkey}{%
1506   \msg{[star addition/#1]}%
1507 }
1508 \AddToKeyvalOption*{testdefaultkey}{%
1509   \msg{[star addition/#1]}%
1510 }
1511 \setkeys{testfamily}{%
1512   testkey=testA,%
1513   testdefaultkey=testB,%
1514   testdefaultkey,%
1515 }
1516 \@@end
1517 ⟨/test3⟩
1518 ⟨*test4pkg⟩
1519 \NeedsTeXFormat{LaTeX2e}
1520 \ProvidesPackage{kvoptions-test4}[2010/12/23 package for testing]
1521 \RequirePackage{kvoptions}[2010/12/23]
1522 \SetupKeyvalOptions{%
1523   family=FOO,%
1524   prefix=foo,%
1525   setkeys=\kvsetkeys,%
1526 }
1527 \DeclareStringOption{str}
1528 \define@key{FOO}{set}{%
1529   \setkeys{BAR}{strbar={#1}}%
1530 }
1531 \define@key{BAR}{strbar}{%
1532   \def\foostr{[BAR:#1]}%
1533 }
1534 \ProcessKeyvalOptions*
1535 ⟨/test4pkg⟩
1536 ⟨*test4⟩
1537 \NeedsTeXFormat{LaTeX2e}
1538 \ProvidesFile{kvoptions-test4.tex}[2010/12/23 test file]
1539 \RequirePackage[%
1540   str=A,set=B,str=C,%
1541 ]{kvoptions-test4}[2010/12/23]
1542 \def\TestExpected{C}
1543 \ifx\foostr\TestExpected
1544   \typeout{* Test ok.}%
1545 \else
1546   \typeout{* Result: [\foostr]}%
1547   \typeout{* Expected: [\TestExpected]}%
1548   \errmessage{Test failed!}%
1549 \fi
1550 \csname @@end\endcsname\end
1551 ⟨/test4⟩
```

8 Installation

8.1 Download

Package. This package is available on CTAN[1]:

CTAN:macros/latex/contrib/oberdiek/kvoptions.dtx The source file.

CTAN:macros/latex/contrib/oberdiek/kvoptions.pdf Documentation.

---
[1] ftp://ftp.ctan.org/tex-archive/

Bundle. All the packages of the bundle 'oberdiek' are also available in a TDS compliant ZIP archive. There the packages are already unpacked and the documentation files are generated. The files and directories obey the TDS standard.

CTAN:install/macros/latex/contrib/oberdiek.tds.zip

TDS refers to the standard "A Directory Structure for TeX Files" (CTAN:tds/tds.pdf). Directories with `texmf` in their name are usually organized this way.

8.2 Bundle installation

Unpacking. Unpack the `oberdiek.tds.zip` in the TDS tree (also known as `texmf` tree) of your choice. Example (linux):

```
unzip oberdiek.tds.zip -d ~/texmf
```

Script installation. Check the directory `TDS:scripts/oberdiek/` for scripts that need further installation steps. Package attachfile2 comes with the Perl script `pdfatfi.pl` that should be installed in such a way that it can be called as `pdfatfi`. Example (linux):

```
chmod +x scripts/oberdiek/pdfatfi.pl
cp scripts/oberdiek/pdfatfi.pl /usr/local/bin/
```

8.3 Package installation

Unpacking. The `.dtx` file is a self-extracting docstrip archive. The files are extracted by running the `.dtx` through plain TeX:

```
tex kvoptions.dtx
```

TDS. Now the different files must be moved into the different directories in your installation TDS tree (also known as `texmf` tree):

```
kvoptions.sty              → tex/latex/oberdiek/kvoptions.sty
kvoptions-patch.sty        → tex/latex/oberdiek/kvoptions-patch.sty
kvoptions.pdf              → doc/latex/oberdiek/kvoptions.pdf
example-mycolorsetup.sty → doc/latex/oberdiek/example-mycolorsetup.sty
test/kvoptions-test1.tex → doc/latex/oberdiek/test/kvoptions-test1.tex
test/kvoptions-test2.tex → doc/latex/oberdiek/test/kvoptions-test2.tex
test/kvoptions-test3.tex → doc/latex/oberdiek/test/kvoptions-test3.tex
test/kvoptions-test4.tex → doc/latex/oberdiek/test/kvoptions-test4.tex
test/kvoptions-test4.sty → doc/latex/oberdiek/test/kvoptions-test4.sty
kvoptions.dtx              → source/latex/oberdiek/kvoptions.dtx
```

If you have a `docstrip.cfg` that configures and enables docstrip's TDS installing feature, then some files can already be in the right place, see the documentation of docstrip.

8.4 Refresh file name databases

If your TeX distribution (teTeX, mikTeX, . . .) relies on file name databases, you must refresh these. For example, teTeX users run `texhash` or `mktexlsr`.

8.5 Some details for the interested

Attached source. The PDF documentation on CTAN also includes the `.dtx` source file. It can be extracted by AcrobatReader 6 or higher. Another option is pdftk, e.g. unpack the file into the current directory:

```
pdftk kvoptions.pdf unpack_files output .
```

Unpacking with LATEX. The `.dtx` chooses its action depending on the format:

plain TEX: Run docstrip and extract the files.

LATEX: Generate the documentation.

If you insist on using LATEX for docstrip (really, docstrip does not need LATEX), then inform the autodetect routine about your intention:

```
latex \let\install=y\input{kvoptions.dtx}
```

Do not forget to quote the argument according to the demands of your shell.

Generating the documentation. You can use both the `.dtx` or the `.drv` to generate the documentation. The process can be configured by the configuration file `ltxdoc.cfg`. For instance, put this line into this file, if you want to have A4 as paper format:

```
\PassOptionsToClass{a4paper}{article}
```

An example follows how to generate the documentation with pdfLATEX:

```
pdflatex kvoptions.dtx
makeindex -s gind.ist kvoptions.idx
pdflatex kvoptions.dtx
makeindex -s gind.ist kvoptions.idx
pdflatex kvoptions.dtx
```

9 References

[1] A guide to key-value methods, Joseph Wright, second draft for TUG-Boat, 2009-03-17. http://www.texdev.net/wp-content/uploads/2009/03/keyval.pdf

[2] Package ifthen, David Carlisle, 2001/05/26. CTAN:macros/latex/base/ifthen.dtx

[3] Package helvet, Sebastian Rahtz, Walter Schmidt, 2004/01/26. CTAN:macros/latex/required/psnfss/psfonts.dtx

[4] Package hyperref, Sebastian Rahtz, Heiko Oberdiek, 2006/02/12. CTAN:macros/latex/contrib/hyperref/

[5] Package keyval, David Carlisle, 1999/03/16. CTAN:macros/latex/required/graphics/keyval.dtx

[6] Package multicol, Frank Mittelbach, 2004/02/14. CTAN:macros/latex/required/tools/multicol.dtx

[7] Package tabularx, David Carlisle, 1999/01/07. CTAN:macros/latex/required/tools/tabularx.dtx

[8] Package tracefnt, Frank Mittelbach, Rainer Schöpf, 1997/05/29. CTAN:macros/latex/base/ltfsstrc.dtx

[9] Package xkeyval, Hendri Adriaens, 2005/05/07. CTAN:macros/latex/contrib/xkeyval/

[10] The LATEX3 Project, *LATEX 2ε for class and package writers*, 2003/12/09. CTAN:macros/latex/doc/clsguide.pdf

10 History

[0000/00/00 v0.0]

- Probably David Carlisle's code in hyperref was the start.

[2004/02/22 v1.0]

- The first version was never published. It also has offered a patch to get rid of LaTeX's option expansion.

[2006/02/16 v2.0]

- Now the package is redesigned with an easier user interface.

- \ProcessKeyvalOptions remains the central service, inherited from hyperref's \ProcessOptionsWithKV. Now the use inside classes is also supported.

- Provides help macros for boolean and simple string options.

- Fixes for the patch of LaTeX. The patch is only enabled, if the user requests it.

[2006/02/20 v2.1]

- Unused option list is sanitized to prevent problems with other packages that uses own processing methods for key value options. Disadvantage: the unused global option detection is weakened.

- New option type by \DeclareVoidOption for options without value.

- Default rule by \DeclareDefaultOption.

- Dynamic options: \DisableKeyvalOption.

[2006/06/01 v2.2]

- Fixes for option patch.

[2006/08/17 v2.3]

- \DeclareBooleanOption renamed to \DeclareBoolOption to avoid a name clash with package \ifoption.

[2006/08/22 v2.4]

- Option patch: \ExecuteOptions does not change the meaning of macro \CurrentOption at all.

[2007/04/11 v2.5]

- Line ends sanitized.

[2007/05/06 v2.6]

- Uses package etexcmds.

[2007/06/11 v2.7]

- The patch part fixes LaTeX bug latex/3965.

[2007/10/02 v2.8]

- Compatibility for plain TeX added.

- Typos in documentation fixed (Axel Sommerfeldt).

[2007/10/11 v2.9]

- Bug fix for option `patch`.

[2007/10/18 v3.0]

- New package kvoptions-patch.

[2009/04/10 v3.1]

- Space by line end removed in definition of internal macro.

[2009/07/17 v3.2]

- `\ProcessLocalKeyvalOptions` added.

- `\DisableKeyvalOption` with the `action=ignore` option fixed (Joseph Wright).

[2009/07/21 v3.3]

- `\DeclareLocalOption`, `\DeclareLocalOptions` added.

[2009/08/13 v3.4]

- Documentation addition: recommendation for Joseph Wright's review article.

- Documentation addition: local/global options.

[2009/12/04 v3.5]

- `\AddToKeyvalOption` added.

[2009/12/08 v3.6]

- Fix: If a default handler is configured, it is now also called for classes.

[2010/02/22 v3.7]

- Missing space in error message added.

[2010/07/23 v3.8]

- Documenation for package kvoptions-patch improved. No code changes.

[2010/12/02 v3.9]

- Key `setkeys` added for `\SetupKeyvalOptions`.

[2010/12/23 v3.10]

- `\DeclareVoidOption` also parses the second parameter as TeX argument to improve compatibility with `\DeclareOption`.

11 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; plain numbers refer to the code lines where the entry is used.

46