

The kvoptions package

Heiko Oberdiek
<oberdiek@uni-freiburg.de>

2007/10/02 v2.8

Abstract

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

Contents

1	Introduction	2
1.1	The beginning	2
1.2	Overview	3
2	Usage	3
2.1	Process options	3
2.1.1	\ProcessKeyvalOptions	3
2.1.2	\SetupKeyvalOptions	3
2.2	Option declarations	4
2.2.1	\DeclareStringOption	4
2.2.2	\DeclareBoolOption	4
2.2.3	\DeclareComplementaryOption	5
2.2.4	\DeclareVoidOption	5
2.2.5	\DeclareDefaultOption	6
2.2.6	Dynamic options	6
2.2.7	\DisableKeyvalOption	6
2.3	Summary of internal macros	7
2.4	plain-TeX	7
3	Example	8
4	Package options	9
4.1	Option patch	9
4.2	Option debugshow	11
5	Limitations	11
5.1	Compatibility	11
5.1.1	Option patch vs. package xkvtxp	11
5.2	Limitations	11
5.2.1	Option comparisons	11
5.2.2	Option list parsing with option patch	12
6	Implementation	12
6.1	Preamble	12
6.2	Option declaration macros	14
6.2.1	\SetupKeyvalOptions	14
6.2.2	\DeclareBoolOption	15
6.2.3	\DeclareStringOption	17
6.2.4	\DeclareVoidOption	18

6.2.5	<code>\DeclareDefaultOption</code>	19
6.3	Dynamic options	19
6.3.1	<code>\DisableKeyvalOption</code>	19
6.4	Process options	21
6.5	<code>\ProcessKeyvalOptions</code>	21
6.5.1	Helper macros	23
6.6	plain- <code>T_EX</code>	24
6.7	Patch	24
7	Test	31
7.1	Preface for standard catcode check	31
7.2	Catcode checks for loading	31
8	Installation	32
8.1	Download	32
8.2	Bundle installation	32
8.3	Package installation	32
8.4	Refresh file name databases	33
8.5	Some details for the interested	33
9	References	33
10	History	34
	[0000/00/00 v0.0]	34
	[2004/02/22 v1.0]	34
	[2006/02/16 v2.0]	34
	[2006/02/20 v2.1]	34
	[2006/06/01 v2.2]	34
	[2006/08/17 v2.3]	34
	[2006/08/22 v2.4]	34
	[2007/04/11 v2.5]	34
	[2007/05/06 v2.6]	35
	[2007/06/11 v2.7]	35
	[2007/10/02 v2.8]	35
11	Index	35

1 Introduction

1.1 The beginning

This package 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 L^AT_EX's package and class `options`:

Package <code>keyval</code>	Package <code>kvoptions</code>	L ^A T _E X kernel
<code>\define@key</code>	<code>\DeclareVoidOption</code> <code>\DeclareStringOption</code> <code>\DeclareBoolOption</code> <code>\DeclareComplementaryOption</code> <code>\DisableKeyvalOption</code>	<code>\DeclareOption</code>
	<code>\DeclareDefaultOption</code>	<code>\DeclareOption*</code>
	<code>\ProcessKeyvalOptions</code>	<code>\ProcessOptions*</code>
	Option patch	Class/package option system
	<code>\SetupKeyvalOptions</code>	

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 L^AT_EX'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.

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

2.1.2 `\SetupKeyvalOptions`

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

This command allows to configure the default assumptions that are based on the current package or class name. L^AT_EX 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	<code>\@currname</code>	(foobar)	<code>\ProcessKeyvalOptions*</code> <code>\DeclareBoolOption</code> <code>\DeclareStringOption</code>
prefix	<code>\@currname</code> @	(foobar@)	<code>\DeclareBoolOption</code> <code>\DeclareStringOption</code> <code>\DeclareVoidOption</code>

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 2.1.2). The initial contents of the macro can be given by the first optional argument *<init>*. The default is empty.

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` complains 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 respectively. 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.:

```
\iffobar@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 be 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 2.1.2). 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 L^AT_EX'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 L^AT_EX gives for unknown 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 L^AT_EX do the job and just call `\OptionNotUsed`. Or the options can be passed to another class that is later loaded.

2.2.6 Dynamic options

Options of L^AT_EX'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.

2.2.7 \DisableKeyvalOption

`\DisableKeyvalOption [⟨options⟩] {⟨family⟩} {⟨key⟩}`

`⟨options⟩:`

<code>action</code>	<code>= undef, warning, error, or ignore</code>	default: <code>undef</code>
<code>global or local</code>		default: <code>global</code>
<code>package or class = ⟨name⟩</code>		

`\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 exclusively 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 `\DisableKeyvalOptions` can be executed everywhere, also outside the package. Therefore the family name and the package/class name is usually unknown for `\DisableKeyvalOptions`. 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}%
}

\iffobar@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.
```

2.3 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 `\langle prefix \rangle` that can be configured by `\SetupKeyvalOptions`.

Declare <code>\langle key \rangle</code>	Defined macro	Description
<code>\DeclareStringOption</code>	<code>\langle prefix \rangle \langle key \rangle</code>	holds the string
<code>\DeclareBoolOption</code>	<code>\if \langle prefix \rangle \langle key \rangle</code> <code>\langle prefix \rangle \langle key \rangle false</code> <code>\langle prefix \rangle \langle key \rangle true</code>	boolean switch disable switch enable switch
<code>\DeclareComplementaryOption</code>	<code>\langle prefix \rangle \langle key \rangle false</code> <code>\langle prefix \rangle \langle key \rangle true</code>	enable parent switch disable parent switch
<code>\DeclareVoidOption</code>	<code>\langle prefix \rangle \langle key \rangle</code>	holds the action

2.4 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}[2007/10/02 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 options 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 messages 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 options.

4.1 Option `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 extremely high. \LaTeX expands the option. 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[pdauthor=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[pdauthor=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 option `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[patch]{kvoptions}
\documentclass[pdauthor=Fran\c coise M\"uller]{article}
\usepackage{hyperref}
```

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

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

Some remarks:

- The patch requires $\varepsilon\text{-TeX}$, 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 $\varepsilon\text{-TeX}$.
- The code for the patch is quite long, there are many test cases. Thus the probability for bugs is probably not too small.

4.2 Option debugshow

The name of this option follows the convention of packages `multicol`, `tabularx`, and `tracefmt`. 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 Option patch vs. package xkvltxp

Package `xkvltxp` from the `xkeyval` project has the same goal as `kvoptions`'s option `patch` and patches \LaTeX 's kernel commands. Of course they cannot be used both. The user must decide, which method he prefers. Package `kvoptions` drops option `patch` if it detects that `xkvltxp` is already loaded.

However both *packages* can be used together, example:

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

The other way should work, too.

Package `kvoptions` 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[patch]{kvoptions}
\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 option patch

With option `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[patch]{kvoptions}
\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 L^AT_EX.

```
99 \begingroup
100 \catcode44 12 % ,
101 \catcode45 12 % -
102 \catcode46 12 % .
103 \catcode58 12 % :
104 \catcode64 11 % @
105 \expandafter\let\expandafter\x\csname ver@kvoptions.sty\endcsname
106 \ifcase 0%
107   \ifx\x\relax % plain
108   \else
109     \ifx\x\empty % LaTeX
110     \else
111       1%
112     \fi
113   \fi
114 \else
115   \expandafter\ifx\csname PackageInfo\endcsname\relax
116     \def\x#1#2{%
117       \immediate\write-1{Package #1 Info: #2.}%
118     }%
119   \else
120     \def\x#1#2{\PackageInfo{#1}{#2, stopped}}%
121   \fi
122   \x{kvoptions}{The package is already loaded}%
123 \endgroup
124 \expandafter\endinput
125 \fi
126 \endgroup
```

Package identification:

```
127 \begingroup
```

```

128 \catcode40 12 % (
129 \catcode41 12 % )
130 \catcode44 12 % ,
131 \catcode45 12 % -
132 \catcode46 12 % .
133 \catcode47 12 % /
134 \catcode58 12 % :
135 \catcode64 11 % @
136 \expandafter\ifx\csname ProvidesPackage\endcsname\relax
137   \def\x#1#2#3[#4]{\endgroup
138     \immediate\write-1{Package: #3 #4}%
139     \xdef#1{#4}%
140   }%
141 \else
142   \def\x#1#2[#3]{\endgroup
143     #2[{#3}]%
144     \ifx#1\relax
145       \xdef#1{#3}%
146     \fi
147   }%
148 \fi
149 \expandafter\x\csname ver@kvoptions.sty\endcsname
150 \ProvidesPackage{kvoptions}%
151 [2007/10/02 v2.8 Connects package keyval with LaTeX options (H0)]

```

Catcodes

```

152 \expandafter\edef\csname KVO@AtEnd\endcsname{%
153   \catcode64 \the\catcode64\relax
154 }
155 \catcode64 11 % @
156 \def\TMP@EnsureCode#1#2{%
157   \edef\KVO@AtEnd{%
158     \KVO@AtEnd
159     \catcode#1 \the\catcode#1\relax
160   }%
161   \catcode#1 #2\relax
162 }
163 \TMP@EnsureCode{1}{14}% ^^A (comment)
164 \TMP@EnsureCode{2}{14}% ^^A (comment)
165 \TMP@EnsureCode{33}{12}% !
166 \TMP@EnsureCode{39}{12}% '
167 \TMP@EnsureCode{40}{12}% (
168 \TMP@EnsureCode{41}{12}% )
169 \TMP@EnsureCode{42}{12}% *
170 \TMP@EnsureCode{44}{12}% ,
171 \TMP@EnsureCode{46}{12}% .
172 \TMP@EnsureCode{47}{12}% /
173 \TMP@EnsureCode{58}{12}% :
174 \TMP@EnsureCode{61}{12}% =
175 \TMP@EnsureCode{62}{12}% >
176 \TMP@EnsureCode{94}{7}% ^ (superscript)
177 \TMP@EnsureCode{96}{12}% ‘

```

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`.

```

178 \ifundefined{define@key}{%
179   \RequirePackage{keyval}\relax
180 }{}

```

Provide macros for plain-TeX.

```

181 \@ifundefined{@onelevel@sanitize}{%
182   \def\@onelevel@sanitize#1{%
183     \edef#1{\expandafter\strip@prefix\meaning#1}%
184   }%
185 }{}
186 \@ifundefined{strip@prefix}{%
187   \def\strip@prefix#1>{}%
188 }{}
189 \@ifundefined{@x@protect}{%
190   \def\@x@protect#1\fi#2#3{%
191     \fi\protect#1%
192   }%
193   \let\@typeset@protect\relax
194 }{}
195 \@ifundefined{@currname}{%
196   \def\@currname{}%
197 }{}
198 \@ifundefined{@currentx}{%
199   \def\@currentx{}%
200 }{}

```

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

```

201 \DeclareOption{debugshow}{\catcode\@ne=9 }
202 \DeclareOption{patch}{\catcode\tw@=9 }

Optionen auswerten:
203 \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.

`\KV0@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 L^AT_EX's class/package system.

```

204 \define@key{KV0}{family}{%
205   \expandafter\edef\csname KV0@family@%
206     \@currname.\@currentx\endcsname{#1}%
207 }
208 \def\KV0@family{%
209   \@ifundefined{KV0@family@\@currname.\@currentx}{%
210     \@currname
211   }{%
212     \csname KV0@family@\@currname.\@currentx\endcsname
213   }%
214 }

```

`\KV0@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.

```

215 \define@key{KV0}{prefix}{%
216   \expandafter\edef\csname KV0@prefix@%

```

```

217     \@currname.\@current\endcsname{#1}%
218 }
219 \def\KV0@prefix{%
220   \@ifundefined{KV0@prefix@\@currname.\@current}{%
221     \@currname %
222   }{%
223     \csname KV0@prefix@\@currname.\@current\endcsname
224   }%
225 }

```

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

```

226 \newcommand*\SetupKeyvalOptions{%
227   \setkeys{KV0}%
228 }

```

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.

L^AT_EX's `\newif` does not check for already defined macros, therefore we add this check here to prevent the user from accidentally redefining of T_EX's primitives and other macros.

```

229 \newcommand*\DeclareBoolOption}[2][false]{%
230   \KV0@ifdefinable{if\KV0@prefix#2}{%
231     \KV0@ifdefinable{\KV0@prefix#2true}{%
232       \KV0@ifdefinable{\KV0@prefix#2false}{%
233         \csname newif\expandafter\endcsname
234         \csname if\KV0@prefix#2\endcsname
235         \@ifundefined{\KV0@prefix#2#1}{%
236           \PackageWarning{kvoptions}{%
237             Initialization of option ‘#2’ failed,\MessageBreak
238             cannot set boolean option to ‘#1’,\MessageBreak
239             use ‘true’ or ‘false’, now using ‘false’%
240           }%
241         }{%
242           \csname\KV0@prefix#2#1\endcsname
243         }%
244         \begingroup
245         \edef\x{\endgroup
246           \noexpand\define@key{\KV0@family}{#2}[true]{%
247             \noexpand\KV0@boolkey{\@currname}%
248             \ifx\@current\@clsextension
249               \noexpand\@clsextension
250             \else
251               \noexpand\@pkgextension
252             \fi
253             {\KV0@prefix}{#2}{####1}%
254           }%
255         }%
256         \x
257       }%
258     }%
259   }%
260 }

```

\DeclareComplementaryOption 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.

```

261 \newcommand*{\DeclareComplementaryOption}[2]{%
262   \@ifundefined{ifKV0@prefix#2}{%
263     \PackageError{kvoptions}{%
264       Cannot generate option code for ‘#1’,\MessageBreak
265       parent switch ‘#2’ does not exist%
266     }{%
267       You are inside %
268       \ifx\@currentx\@clsextension class\else package\fi\space
269       ‘\@currname.\@currentx’.\MessageBreak
270       ‘KV0@family’ is used as familiy %
271       for the keyval options.\MessageBreak
272       ‘KV0@prefix’ serves as prefix %
273       for internal switch macros.\MessageBreak
274       \MessageBreak
275       \@ehc
276     }%
277   }{%
278     \KV0@ifdefinable{\KV0@prefix#1true}{%
279       \KV0@ifdefinable{\KV0@prefix#1false}{%
280         \expandafter\let\csname\KV0@prefix#1false\expandafter\endcsname
281         \csname\KV0@prefix#2true\endcsname
282         \expandafter\let\csname\KV0@prefix#1true\expandafter\endcsname
283         \csname\KV0@prefix#2false\endcsname

```

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

```

284   \begingroup
285   \edef\x{\endgroup
286     \noexpand\define@key{\KV0@family}{#1}[true]{%
287       \noexpand\KV0@boolkey{\@currname}%
288       \ifx\@currentx\@clsextension
289         \noexpand\@clsextension
290       \else
291         \noexpand\@pkgextension
292       \fi
293       {\KV0@prefix}{#1}{####1}%
294     }%
295   }%
296   \x
297 }%
298 }%
299 }%
300 }

```

`\KV0@ifdefinable` Generate the command token LaTeX’s `\@ifdefinable` expects.

```

301 \def\KV0@ifdefinable#1{%
302   \expandafter\@ifdefinable\csname #1\endcsname
303 }

```

`\KV0@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

```

```

304 \def\KV0@boolkey#1#2#3#4#5{%
305   \edef\KV0@param{#5}%
306   \@onelevel@sanitize\KV0@param
307   \ifx\KV0@param\KV0@true

```

```

308     \expandafter\@firstofone
309 \else
310     \ifx\KVO@param\KVO>false
311     \expandafter\expandafter\expandafter\@firstofone
312 \else
313     \ifx#2\@clsextension
314     \expandafter\ClassWarning
315 \else
316     \expandafter\PackageWarning
317 \fi
318     {#1}{%
319         Value '\KVO@param' is not supported by\MessageBreak
320         option '#4'%
321     }%
322     \expandafter\expandafter\expandafter\@gobble
323 \fi
324 \fi
325 {%
326     ^^A\ifx#2\@clsextension
327     ^^A \expandafter\ClassInfo
328     ^^A\else
329     ^^A \expandafter\PackageInfo
330     ^^A\fi
331     ^^A{#1}{[option] #4=\KVO@param}%
332     \csname#3#4\KVO@param\endcsname
333 }%
334 }

```

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

```

335 \def\KVO>true{true}
336 \def\KVO>false{false}
337 \@onelevel@sanitize\KVO>true
338 \@onelevel@sanitize\KVO>false

```

6.2.3 \DeclareStringOption

\DeclareStringOption

```

339 \newcommand*{\DeclareStringOption}[2][]{%
340     \@ifnextchar[%
341         \KVO@DeclareStringOption{#1}{#2}%
342     ]{%
343         \KVO@DeclareStringOption{#1}{#2}{}%
344     }%
345 }

```

\KVO@DeclareStringOption

```

346 \def\KVO@DeclareStringOption#1#2#3[#4]{%
347     \KVO@ifdefinable{\KVO@prefix#2}{%
348         \@namedef{\KVO@prefix#2}{#1}%
349         \begingroup
350             \ifx\#3\%
351                 \toks@{%
352                     \else
353                     \toks@{[#4]}%
354                 \fi
355                 \edef\x{\endgroup
356                     \noexpand\define@key{\KVO@family}{#2}\the\toks@{%
357                         ^^A\begingroup
358                         ^^A \toks@{####1}%
359                         ^^A \ifx\@current\@clsextension
360                         ^^A \noexpand\ClassInfo

```

```

361      ^^A \else
362      ^^A \noexpand\PackageInfo
363      ^^A \fi
364      ^^A {\@currname}{%
365      ^^A [option] #2={\noexpand\the\toks@}%
366      ^^A }%
367      ^^A\endgroup
368      \noexpand\def
369      \expandafter\noexpand\csname\KVO@prefix#2\endcsname{####1}%
370      }%
371  }%
372  \x
373 }%
374 }

```

6.2.4 \DeclareVoidOption

\DeclareVoidOption

```

375 \newcommand*{\DeclareVoidOption}[1]{%
376   \begingroup
377   \let\next\@gobbletwo
378   \KVO@ifdefinable{\KVO@prefix#1}{%
379     \let\next\@firstofone
380   }%
381   \expandafter\endgroup
382   \next{%
383     \begingroup
384     \edef\x{\endgroup
385       \noexpand\define@key{\KVO@family}{#1}[\KVO@VOID@]{%
386         \noexpand\KVO@voidkey{\@currname}%
387         \ifx\@current\@clsextension
388           \noexpand\@clsextension
389         \else
390           \noexpand\@pkgextension
391         \fi
392         {#1}%
393         {####1}%
394         \expandafter\noexpand\csname\KVO@prefix#1\endcsname
395       }%
396     }%
397     \x
398     \@namedef{\KVO@prefix#1}%
399   }%
400 }
401 \def\KVO@VOID@{@VOID@}

```

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

```

402 \def\KVO@voidkey#1#2#3#4{%
403   \def\CurrentOption{#3}%
404   \begingroup
405     \def\x{#4}%
406   \expandafter\endgroup
407   \ifx\x\KVO@VOID@
408   \else
409     \ifx#2\@clsextension
410       \expandafter\ClassWarning
411     \else
412       \expandafter\PackageWarning

```

```

413 \fi
414 {#1}{%
415     Unexpected value for option ‘#3’\MessageBreak
416     is ignored%
417 }%
418 \fi
419 ^^A\ifx#2\@clsextension
420 ^^A \expandafter\ClassInfo
421 ^^A\else
422 ^^A \expandafter\PackageInfo
423 ^^A\fi
424 ^^A{#1}{[option] #3}%
425 }

```

6.2.5 \DeclareDefaultOption

\DeclareDefaultOption

```

426 \newcommand*{\DeclareDefaultOption}{%
427 \@namedef{KV0@default@\@currname.\@currentx}%
428 }

```

6.3 Dynamic options

6.3.1 \DisableKeyvalOption

```

429 \SetupKeyvalOptions{%
430     family=KV0dyn,%
431     prefix=KV0dyn@%
432 }
433 \DeclareBoolOption[true]{global}
434 \DeclareComplementaryOption{local}{global}
435 \DeclareStringOption[undef]{action}
436 \let\KV0dyn@name\relax
437 \let\KV0dyn@ext\@empty
438 \define@key{KV0dyn}{class}{%
439     \def\KV0dyn@name{#1}%
440     \let\KV0dyn@ext\@clsextension
441 }
442 \define@key{KV0dyn}{package}{%
443     \def\KV0dyn@name{#1}%
444     \let\KV0dyn@ext\@pkgextension
445 }
446 \newcommand*{\DisableKeyvalOption}[3][[]]{%
447     \begingroup
448         \setkeys{KV0dyn}{#1}%
449         \def\x{\endgroup}%
450         \@ifundefined{KV0@action@\KV0dyn@action}{%
451             \PackageError{kvoptions}{%
452                 Unknown disable action %
453                 ‘\expandafter\strip@prefix\meaning\KV0dyn@action’\MessageBreak
454                 for option ‘#3’ in keyval family ‘#2’%
455             }\@ehc
456         }{%
457             \csname KV0@action@\KV0dyn@action\endcsname{#2}{#3}%
458         }%
459     \x
460 }
461 \def\KV0@action@undef#1#2{%
462     \edef\x{\endgroup
463         \ifKV0dyn@global\global\fi
464         \let
465         \expandafter\noexpand\csname KV@#1@#2\endcsname

```

```

466 \relax
467 \ifKV0dyn@global\global\fi
468 \let
469 \expandafter\noexpand\csname KV@#1@#2@default\endcsname
470 \relax
471 }%
472 ^^A\PackageInfo{kvoptions}{%
473 ^^A [option] key '#2' of family '#1'\MessageBreak
474 ^^A is disabled (undef, \ifKV0dyn@global\global\else local\fi)%
475 ^^A}%
476 }
477 \def\KV0@action@ignore#1#2{%
478 \edef\x{\endgroup
479 \ifKV0dyn@global\global\fi
480 \let
481 \expandafter\noexpand\csname KV@#1@#2\endcsname
482 \@gobble
483 \ifKV0dyn@global\global\fi
484 \let
485 \expandafter\noexpand\csname KV@#1@#2@default\endcsname
486 \@empty
487 }%
488 ^^A\PackageInfo{kvoptions}{%
489 ^^A [option] key '#2' of family '#1'\MessageBreak
490 ^^A is disabled (ignore, \ifKV0dyn@global\global\else local\fi)%
491 ^^A}%
492 }
493 \def\KV0@action@error{%
494 \KV0@do@action{error}%
495 }
496 \def\KV0@action@warning{%
497 \KV0@do@action{warning}%
498 }

#1 error or warning
#2 <family>
#3 <key>
499 \def\KV0@do@action#1#2#3{%
500 \ifx\KV0dyn@name\relax
501 \PackageError{kvoptions}{%
502 Action type '#1' needs package/class name\MessageBreak
503 for key '#3' in family '#2'%
504 }\@ehc
505 \else
506 \edef\x{\endgroup
507 \noexpand\define@key{#2}{#3}[]{%
508 \expandafter\noexpand\csname KV0@disable@#1\endcsname
509 {\KV0dyn@name}\noexpand\KV0dyn@ext{#3}%
510 }%
511 \ifKV0dyn@global
512 \global\let
513 \expandafter\noexpand\csname KV@#2@#3\endcsname
514 \expandafter\noexpand\csname KV@#2@#3\endcsname
515 \global\let
516 \expandafter\noexpand\csname KV@#2@#3@default\endcsname
517 \expandafter\noexpand\csname KV@#2@#3@default\endcsname
518 \fi
519 }%
520 ^^A\ifx\KV0dyn@ext\@clsextension
521 ^^A \expandafter\ClassInfo
522 ^^A\else
523 ^^A \expandafter\PackageInfo
524 ^^A\fi

```

```

525    ^^A{\KV0dyn@name}{%
526    ^^A [option] key ‘#3’ of family ‘#2’\MessageBreak
527    ^^A is disabled (#1, \ifKV0dyn@global global\else local\fi)%
528    ^^A}%
529    \fi
530 }
531 \def\KV0@disable@error#1#2#3{%
532   \ifx#2\@clsextension
533     \expandafter\ClassError
534   \else
535     \expandafter\PackageError
536   \fi
537   {#1}{%
538     Option ‘#3’ is given too late,\MessageBreak
539     now the option is ignored%
540   }\@ehc
541 }
542 \def\KV0@disable@warning#1#2#3{%
543   \ifx#2\@clsextension
544     \expandafter\ClassWarning
545   \else
546     \expandafter\PackageWarning
547   \fi
548   {#1}{%
549     Option ‘#3’ is already consumed\MessageBreak
550     and has no effect%
551   }%
552 }

```

6.4 Process options

6.5 \ProcessKeyvalOptions

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

```

553 \newcommand*{\ProcessKeyvalOptions}{%
554   \@ifstar{%
555     \begingroup
556     \edef\x{\endgroup
557       \noexpand\KV0@ProcessKeyvalOptions{\KV0@family}%
558     }%
559     \x
560   }%
561   \KV0@ProcessKeyvalOptions
562 }
563 \def\KV0@ProcessKeyvalOptions#1{%
564   \let\@tempc\relax
565   \let\KV0@temp\@empty

```

Add any global options that are known to KV to the start of the list being built in `\KV0@temp` and mark them used (by removing them from the unused option list).

```

566   \ifx\@currentx\@clsextension
567   \else
568     \ifx\@classoptionslist\relax
569     \else
570       \for\KV0@CurrentOption:=\@classoptionslist\do{%
571         \@ifundefined{KV@#1}\expandafter\KV0@getkey
572           \KV0@CurrentOption=\@nil}{%
573       }{%
574         \edef\KV0@temp{%
575           ^^B       \etex@unexpanded\expandafter{%
576             \KV0@temp
577           ^^B       }%
578           ,%

```

```

579 ^^B      \etex@unexpanded\expandafter{%
580           \KV0@CurrentOption
581 ^^B      }%
582           ,%
583           }%
584 ^^B      \@onelevel@sanitize\KV0@CurrentOption
585           \@expandtwoargs\@removeelement\KV0@CurrentOption
586           \@unusedoptionlist\@unusedoptionlist
587           }%
588           }%
589     \fi
590   \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`.

```

591   \begingroup
592     \toks\@tw@{}%
593     \@ifundefined{opt@\@currname.\@currentx}{%
594       \toks@\expandafter{\KV0@temp}%
595     }{%
596       \toks@\expandafter\expandafter\expandafter{%
597         \csname opt@\@currname.\@currentx\endcsname
598       }%
599       \ifx\@currentx\@clsextension
600         \edef\CurrentOption{\the\toks@}%
601         \toks@\expandafter{\KV0@temp}%
602         \@for\CurrentOption:=\CurrentOption\do{%
603           \@ifundefined{%
604             KV@#1@\expandafter\KV0@getkey\CurrentOption=\@nil
605           }{%

```

A class puts not used options in the unused option list.

```

606 ^^B      \@onelevel@sanitize\CurrentOption
607           \ifx\@unusedoptionlist\@empty
608           \global\let\@unusedoptionlist\CurrentOption
609           \else
610             \expandafter\expandafter\expandafter\gdef
611             \expandafter\expandafter\expandafter\@unusedoptionlist
612             \expandafter\expandafter\expandafter{%
613               \expandafter\@unusedoptionlist
614               \expandafter,\CurrentOption
615             }%
616           \fi
617         }{%
618           \toks@\expandafter{%
619             \the\expandafter\toks@\expandafter,\CurrentOption
620           }%
621         }%
622       }%
623     \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.

```

624       \@ifundefined{KV0@default@\@currname.\@currentx}{%
625         \toks@\expandafter\expandafter\expandafter{%
626           \expandafter\KV0@temp\the\toks@
627         }%
628       }{%
629         \edef\CurrentOption{\the\toks@}%
630         \toks@\expandafter{\KV0@temp}%
631         \@for\CurrentOption:=\CurrentOption\do{%
632           \@ifundefined{%

```

```

633         KV@#1@\expandafter\KV0@getkey\CurrentOption=\@nil
634     }{%
635         \toks\tw@\expandafter{%
636             \the\toks\expandafter\tw@\expandafter,\CurrentOption
637         }%
638     }{%
639         \toks@\expandafter{%
640             \the\expandafter\toks@\expandafter,\CurrentOption
641         }%
642     }%
643 }%
644 }%
645 \fi
646 }%
647 \edef\KV0@temp{\endgroup
648     \noexpand\KV0@calldefault{\the\toks\tw@}%
649     \noexpand\setkeys{#1}{\the\toks@}%
650 }%
651 \KV0@temp

```

Some cleanup of \ProcessOptions.

```

652 \let\CurrentOption\@empty
653 \AtEndOfPackage{\let\@unprocessedoptions\relax}%
654 }

```

6.5.1 Helper macros

`\KV0@getkey` Extract the key part of a key=value pair.

```

655 \def\KV0@getkey#1=#2\@nil{#1}

```

`\KV0@calldefault`

```

656 \def\KV0@calldefault#1{%
657     \begingroup
658     \def\x{#1}%
659     \expandafter\endgroup
660     \ifx\x\@empty
661     \else
662         \@for\CurrentOption:=#1\do{%
663             \ifx\CurrentOption\@empty
664             \else
665                 \expandafter\KV0@setcurrents\CurrentOption=\@nil
666                 \@nameuse{KV0@default@\@currname.\@current}%
667             \fi
668         }%
669     \fi
670 }

```

`\KV0@setcurrents` Extract the key part of a key=value pair.

```

671 \def\KV0@setcurrents#1=#2\@nil{
672     \def\CurrentOptionValue{#2}%
673     \ifx\CurrentOptionValue\@empty
674         \let\CurrentOptionKey\CurrentOption
675         \let\CurrentOptionValue\relax
676     \else
677         \edef\CurrentOptionKey{\zap@space#1 \@empty}%
678         \expandafter\KV0@setcurrentvalue\CurrentOption\@nil
679     \fi
680 }

```

`\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.

```

681 \def\KV0@setcurrentvalue#1=#2\@nil{%

```

```

682 \KV@@sp@def\CurrentOptionValue{#2}%
683 }

```

6.6 plain-TeX

Disable L^AT_EX stuff.

```

684 \begingroup\expandafter\expandafter\expandafter\endgroup
685 \expandafter\ifx\csname documentclass\endcsname\relax
686 \def\ProcessKeyvalOptions{%
687 \ifstar{}\@gobble
688 }%
689 \catcode\tw@=14 %
690 \fi

```

6.7 Patch

Caution: docstrip stops at \endinput if it starts a line.

```

691 ^^B\@gobbletwo
692 \KV@@AtEnd\endinput

693 \PackageInfo{kvoptions}{Patching LaTeX's option system}

Check for  $\varepsilon$ -TeX.
694 \begingroup\expandafter\expandafter\expandafter\endgroup
695 \expandafter\ifx\csname eTeXversion\endcsname\relax
696 \PackageWarningNoLine{kvoptions}{%
697 Option 'patch' ignored, because e-TeX is missing%
698 }%
699 \expandafter\endinput
700 \fi

Package etexcmds for \etex@unexpanded.
701 \RequirePackage{etexcmds}
702 \ifetex@unexpanded
703 \else
704 \PackageError{kvoptions}{%
705 Could not find eTeX's \string\unexpanded.\MessageBreak
706 Try adding \string\RequirePackage\string{etexcmds\string} %
707 before \string\documentclass%
708 }\@ehd
709 \expandafter\endinput
710 \fi

Check for package xkvltxp.
711 \ifpackageloaded{xkvltxp}{%
712 \PackageWarningNoLine{kvoptions}{%
713 Option 'patch' cannot be used together with\MessageBreak
714 package 'xkvltxp' that is already loaded.\MessageBreak
715 Therefore option 'patch' will be ignored%
716 }%
717 \endinput
718 }{}

719 \def\@if@options#1#2#3{%
720 \begingroup
721 \KV@@normalize\KV@@temp{#3}%
722 \edef\x{\endgroup
723 \noexpand\@if@pti@ns{%
724 \detokenize\expandafter\expandafter\expandafter{%
725 \csname opt@#2.#1\endcsname
726 }%
727 }{%
728 \detokenize\expandafter{\KV@@temp}%
729 }%

```

```

730 }%
731 \x
732 }

733 \def\@passOptions#1#2#3{%
734 \KV0@normalize\KV0@temp{#2}%
735 \@ifundefined{opt@#3.#1}{%
736 \expandafter\gdef\csname opt@#3.#1%
737 \expandafter\endcsname\expandafter{%
738 \KV0@temp
739 }%
740 }{%
741 \expandafter\gdef\csname opt@#3.#1%
742 \expandafter\expandafter\expandafter\endcsname
743 \expandafter\expandafter\expandafter{%
744 \csname opt@#3.#1\expandafter\endcsname\expandafter,\KV0@temp
745 }%
746 }%
747 }

748 \def\ProcessOptions{%
749 \let\ds@\@empty
750 \@ifundefined{opt@\@currname.\@currentx}{%
751 \let\@curroptions\@empty
752 }{%
753 \expandafter\expandafter\expandafter\def
754 \expandafter\expandafter\expandafter\@curroptions
755 \expandafter\expandafter\expandafter{%
756 \csname opt@\@currname.\@currentx\endcsname
757 }%
758 }%
759 \@ifstar\KV0@xprocessoptions\KV0@processoptions
760 }

761 \def\KV0@processoptions{%
762 \@for\CurrentOption:=\@declaredoptions\do{%
763 \ifx\CurrentOption\@empty
764 \else
765 \begingroup
766 \ifx\@currentx\@clsextension
767 \toks@{}%
768 \else
769 \toks@\expandafter{\@classoptionslist,}%
770 \fi
771 \toks\tw@\expandafter{\@curroptions}%
772 \edef\x{\endgroup
773 \noexpand\in@{\CurrentOption,}{,\the\toks@\the\toks\tw@,}%
774 }%
775 \x
776 \ifin@
777 \KV0@useoption
778 \expandafter\let\csname ds@\CurrentOption\endcsname\@empty
779 \fi
780 \fi
781 }%
782 \KV0@processoptions
783 }

784 \def\KV0@xprocessoptions{%
785 \ifx\@currentx\@clsextension
786 \else
787 \@for\CurrentOption:=\@classoptionslist\do{%
788 \ifx\CurrentOption\@empty
789 \else
790 \KV0@in@\CurrentOption\@declaredoptions

```

```

791         \ifin@
792         \KVO@use@option
793         \expandafter\let\csname ds@\CurrentOption\endcsname\@empty
794     \fi
795 \fi
796 }%
797 \fi
798 \KVO@process@ptions
799 }

800 \def\KVO@in@#1#2{%
801     \in@false
802     \begingroup
803     \@for\x:=#2\do{%
804         \ifx\x#1\relax
805             \in@true
806         \fi
807     }%
808     \edef\x{\endgroup
809         \ifin@
810             \noexpand\in@true
811         \fi
812     }%
813     \x
814 }

815 \def\KVO@process@ptions{%
816     \@for\CurrentOption:=\@curroptions\do{%
817         \ifundefined{ds@\KVO@SanitizedCurrentOption}{%
818             \KVO@use@option
819             \default@ds
820         }%
821         \KVO@use@option
822     }%
823     \@for\CurrentOption:=\@declaredoptions\do{%
824         \expandafter\let\csname ds@\CurrentOption\endcsname\relax
825     }%
826     \let\CurrentOption\@empty
827     \let\@fileswith@ptions\@fileswith@ptions
828     \AtEndOfPackage{\let\@unprocessedoptions\relax}%
829 }

830 \def\KVO@use@option{%
831     \begingroup
832     \edef\x{\endgroup
833         \noexpand\@removeelement{%
834             \detokenize\expandafter{\CurrentOption}%
835         }{%
836             \detokenize\expandafter{\@unusedoptionlist}%
837         }%
838     }%
839     \x\@unusedoptionlist
840     \csname ds@\KVO@SanitizedCurrentOption\endcsname
841 }

842 \def\OptionNotUsed{%
843     \ifx\@current\@clsextension
844         \xdef\@unusedoptionlist{%
845             \ifx\@unusedoptionlist\@empty
846                 \else
847                     \detokenize\expandafter{\@unusedoptionlist,}%
848                 \fi
849             \detokenize\expandafter{\CurrentOption}%
850         }%
851     \fi

```

```

852 }

Variant of \ExecuteOptions that better protects \CurrentOption.

853 \def\CurrentOption@SaveLevel{0}
854 \def\ExecuteOptions{%
855   \expandafter\KVO@ExecuteOptions
856   \csname CurrentOption@CurrentOption@SaveLevel\endcsname
857 }
858 \def\KVO@ExecuteOptions#1#2{%
859   \let#1\CurrentOption
860   \edef\CurrentOption@SaveLevel{%
861     \the\numexpr\CurrentOption@SaveLevel+1%
862   }%
863   \@for\CurrentOption:=#2\do{%
864     \csname ds@\CurrentOption\endcsname
865   }%
866   \edef\CurrentOption@SaveLevel{%
867     \the\numexpr\CurrentOption@SaveLevel-1%
868   }%
869   \let\CurrentOption#1%
870 }

871 \def\KVO@fileswith@ptions#1[#2]#3[#4]{%
872   \ifx#1\@clsextension
873     \ifx\@classoptionslist\relax
874       \KVO@normalize\KVO@temp{#2}%
875       \expandafter\gdef\expandafter\@classoptionslist\expandafter{%
876         \KVO@temp
877       }%
878       \def\reserved@a{%
879         \KVO@onefilewithoptions{#3}[#{2}][#{4}]#1%
880         \@documentclasshook
881       }%
882     \else
883       \def\reserved@a{%
884         \KVO@onefilewithoptions{#3}[#{2}][#{4}]#1%
885       }%
886     \fi
887   \else
888     \begingroup
889     \let\KVO@temp\relax
890     \let\KVO@onefilewithoptions\relax
891     \let\@pkgextension\relax
892     \def\reserved@b##1,{%
893       \ifx\@nil##1\relax
894       \else
895         \ifx\relax##1\relax
896         \else
897           \KVO@onefilewithoptions{##1}[{\KVO@temp}][#{4}]%
898           \@pkgextension
899         \fi
900         \expandafter\reserved@b
901       \fi
902     }%
903     \edef\reserved@a{\zap@space#3 \@empty}%
904     \edef\reserved@a{\expandafter\reserved@b\reserved@a,\@nil,}%
905     \toks@{#2}%
906     \def\KVO@temp{\the\toks@}%
907     \edef\reserved@a{\endgroup \reserved@a}%
908   \fi
909   \reserved@a
910 }

911 \def\KVO@onefilewithoptions#1[#2]#3[#4]{%

```

```

912 \@pushfilename
913 \xdef\@currname{#1}%
914 \global\let\@currentx#4%
915 \expandafter\let\csname\@currname.\@currentx-h@@k\endcsname\@empty
916 \let\CurrentOption\@empty
917 \@resetOptions
918 \makeatletter
919 \def\reserved@a{%
920   \@ifl@aded\@currentx{#1}{%
921     \@ifOptions\@currentx{#1}{#2}{%
922       }{%
923         \begingroup
924         \@ifundefined{opt@#1.\@currentx}{%
925           \def\x{%
926             }{%
927               \edef\x{%
928                 \expandafter\expandafter\expandafter\strip@prefix
929                 \expandafter\meaning\csname opt@#1.\@currentx\endcsname
930               }%
931             }%
932             \def\y{#2}%
933             \edef\y{\expandafter\strip@prefix\meaning\y}%
934             \@latex@error{Option clash for \cls@pkg\space #1}{%
935               The package #1 has already been loaded %
936               with options:\MessageBreak
937               \space\space[\x]\MessageBreak
938               There has now been an attempt to load it %
939               with options:\MessageBreak
940               \space\space[\y]\MessageBreak
941               Adding the global options:\MessageBreak
942               \space\space
943               \x,\y\MessageBreak
944               to your \noexpand\documentclass declaration may fix this.%
945               \MessageBreak
946               Try typing \space <return> \space to proceed.%
947             }%
948           \endgroup
949         }%
950       }{%
951         \@passOptions\@currentx{#2}{#1}%
952         \global\expandafter
953         \let\csname ver@\@currname.\@currentx\endcsname\@empty
954         \InputIfFileExists
955         {\@currname.\@currentx}%
956         {}%
957         {\@missingfileerror\@currname\@currentx}%
958         \let\@unprocessedoptions\@unprocessedoptions
959         \csname\@currname.\@currentx-h@@k\endcsname
960         \expandafter\let\csname\@currname.\@currentx-h@@k\endcsname
961         \undefined
962         \@unprocessedoptions
963       }%
964       \@ifl@ter\@currentx{#1}{#3}{%
965         }{%
966           \@latex@warning@no@line{%
967             You have requested,\on@line, %
968             version\MessageBreak
969             #3' of \cls@pkg\space #1,\MessageBreak
970             but only version\MessageBreak
971             '\csname ver@#1.\@currentx\endcsname'\MessageBreak
972             is available%
973           }%

```

```

974 }%
975 \ifx\@currentx\@clsextension\let\LoadClass\@twoloadclasserror\fi
976 \@popfilename
977 \@reset@options
978 }%
979 \reserved@a
980 }

981 \def\@unknownoptionerror{%
982 \@latex@error{%
983   Unknown option '\KV@SanitizedCurrentOption' %
984   for \@cls@pkg\space'\@currname'%
985 }{%
986   The option '\KV@SanitizedCurrentOption' was not declared in %
987   \@cls@pkg\space'\@currname', perhaps you\MessageBreak
988   misspelled its name. %
989   Try typing \space <return> %
990   \space to proceed.%
991 }%
992 }

993 \def\@unprocessedoptions{%
994 \ifx\@currentx\@pkgextension
995   \@ifundefined{opt@\@currname.\@currentx}{%
996     \let\@curroptions\@empty
997   }{%
998     \expandafter\let\expandafter\@curroptions
999       \csname opt@\@currname.\@currentx\endcsname
1000   }%
1001   \@for\CurrentOption:=\@curroptions\do{%
1002     \ifx\CurrentOption\@empty\else\@unknownoptionerror\fi
1003   }%
1004 \fi
1005 }

1006 \def\KV@SanitizedCurrentOption{%
1007 \expandafter\strip@prefix\meaning\CurrentOption
1008 }

  Normalize option list.

1009 \def\KV@normalize#1#2{%
1010 \let\KV@result\@empty
1011 \KV@splitcomma#2,\@nil
1012 \let#1\KV@result
1013 }
1014 \def\KV@splitcomma#1,#2\@nil{%
1015 \KV@ifempty{#1}{}%
1016 \KV@checkkv#1=\@nil
1017 }%
1018 \KV@ifempty{#2}{%\KV@splitcomma#2\@nil}%
1019 }
1020 \def\KV@ifempty#1{%
1021 \expandafter\ifx\expandafter\\detokenize{#1}\\%
1022 \expandafter\@firstoftwo
1023 \else
1024 \expandafter\@secondoftwo
1025 \fi
1026 }
1027 \def\KV@checkkv#1=#2\@nil{%
1028 \KV@ifempty{#2}{%
1029   % option without value
1030   \edef\KV@x{\zap@space#1 \@empty}%
1031   \ifx\KV@x\@empty
1032     % ignore empty option
1033   \else

```

```

1034      % append to list
1035      \edef\KV0@result{%
1036        \etex@unexpanded\expandafter{\KV0@result},\KV0@x
1037      }%
1038    \fi
1039  }{%
1040    % #1: "key", #2: "value="
1041    % add key part
1042    \edef\KV0@result{%
1043      \etex@unexpanded\expandafter{\KV0@result},%
1044      \zap@space#1 \@empty
1045    }%
1046    \futurelet\@let@token\KV0@checkfirsttok#2 \@nil = \@nil|\KV0@nil
1047  }%
1048 }
1049 \def\KV0@checkfirsttok{%
1050   \ifx\@let@token\bgroup
1051     % no space at start
1052     \expandafter\KV0@removelastspace\expandafter=%
1053     % "<value><spaceopt>= \@nil"
1054   \else
1055     \expandafter\KV0@checkfirstA
1056   \fi
1057 }
1058 \def\KV0@checkfirstA#1 #2\@nil{%
1059   \KV0@ifempty{#2}{%
1060     \KV0@removelastspace=#1 \@nil
1061   }{%
1062     \KV0@ifempty{#1}{%
1063       \KV0@removelastspace=#2\@nil
1064     }{%
1065       \KV0@removelastspace=#1 #2\@nil
1066     }%
1067   }%
1068 }
1069 \def\KV0@removelastspace#1 = \@nil|#2\KV0@nil{%
1070   \KV0@ifempty{#2}{%
1071     \edef\KV0@result{%
1072       \etex@unexpanded\expandafter{\KV0@result}%
1073       \etex@unexpanded\expandafter{\KV0@removegarbage#1\KV0@nil}%
1074     }%
1075   }{%
1076     \edef\KV0@result{%
1077       \etex@unexpanded\expandafter{\KV0@result}%
1078       \etex@unexpanded{#1}%
1079     }%
1080   }%
1081 }
1082 \def\KV0@removegarbage#1= \@nil|#2\KV0@nil{#1}%
    Arguments #1 and #2 are macros.
1083 \def\KV0@removeelement#1#2{%
1084   \begingroup
1085   \toks@={}%
1086   \@for\x:=#2\do{%
1087     \ifx\x\@empty
1088     \else
1089       \ifx\x#1\relax
1090     \else
1091       \edef\t{\the\toks@}%
1092       \ifx\t\@empty
1093     \else
1094       \toks@\expandafter{\the\toks@,}%

```

```

1095         \fi
1096         \toks@\expandafter{\the\expandafter\toks@\x}%
1097     \fi
1098 \fi
1099 }%
1100 \edef\x{\endgroup
1101 \def\noexpand#2{\the\toks@}%
1102 }%
1103 \x
1104 }

1105 \let@@fileswith@pti@ns\KV0@fileswith@pti@ns
1106 \ifx\@fileswith@pti@ns\@badrequireerror
1107 \else
1108 \let\@fileswith@pti@ns\KV0@fileswith@pti@ns
1109 \fi

1110 KV0@AtEnd
1111 \</package>

```

7 Test

7.1 Preface for standard catcode check

```

1112 <*test1>
1113 \input miniltx.tex\relax
1114 </test1>

```

7.2 Catcode checks for loading

```

1115 <*test1>

1116 \catcode'\@=11 %
1117 \def\RestoreCatcodes{}
1118 \count@=0 %
1119 \loop
1120 \edef\RestoreCatcodes{%
1121 \RestoreCatcodes
1122 \catcode\the\count@=\the\catcode\count@\relax
1123 }%
1124 \ifnum\count@<255 %
1125 \advance\count@\@ne
1126 \repeat
1127
1128 \def\RangeCatcodeInvalid#1#2{%
1129 \count@=#1\relax
1130 \loop
1131 \catcode\count@=15 %
1132 \ifnum\count@<#2\relax
1133 \advance\count@\@ne
1134 \repeat
1135 }

1136 \def\Test{%
1137 \RangeCatcodeInvalid{0}{47}%
1138 \RangeCatcodeInvalid{58}{64}%
1139 \RangeCatcodeInvalid{91}{96}%
1140 \RangeCatcodeInvalid{123}{255}%
1141 \catcode'\@=12 %
1142 \catcode'\=0 %
1143 \catcode'\{=1 %
1144 \catcode'\}=2 %
1145 \catcode'\#=6 %
1146 \catcode'\[=12 %
1147 \catcode'\]=12 %

```

```

1148 \catcode'\%=14 %
1149 \catcode'\ =10 %
1150 \catcode13=5 %
1151 \input kvoptions.sty\relax
1152 \RestoreCatcodes
1153 }
1154 \Test
1155 \csname @@end\endcsname
1156 \end
1157 </test1>

```

8 Installation

8.1 Download

Package. This package is available on CTAN¹:

[CTAN:macros/latex/contrib/oberdiek/kvoptions.dtx](#) The source file.

[CTAN:macros/latex/contrib/oberdiek/kvoptions.pdf](#) Documentation.

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:macros/latex/contrib/oberdiek/oberdiek-tds.zip](#)

TDS refers to the standard “A Directory Structure for T_EX 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-T_EX:

```
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):

<code>kvoptions.sty</code>	→ <code>tex/latex/oberdiek/kvoptions.sty</code>
<code>kvoptions.pdf</code>	→ <code>doc/latex/oberdiek/kvoptions.pdf</code>
<code>example-mycolorsetup.sty</code>	→ <code>doc/latex/oberdiek/example-mycolorsetup.sty</code>
<code>kvoptions-test1.tex</code>	→ <code>doc/latex/oberdiek/kvoptions-test1.tex</code>
<code>kvoptions.dtx</code>	→ <code>source/latex/oberdiek/kvoptions.dtx</code>

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`.

¹<http://ftp.ctan.org/tex-archive/>

8.4 Refresh file name databases

If your \TeX distribution (te\TeX , mik\TeX , ...) relies on file name databases, you must refresh these. For example, te\TeX users run `texhash` or `mktextlsr`.

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 `pdf \LaTeX` :

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

9 References

- [1] Package `ifthen`, David Carlisle, 2001/05/26.[CTAN:macros/latex/base/ifthen.dtx](#)
- [2] Package `helvet`, Sebastian Rahtz, Walter Schmidt, 2004/01/26.[CTAN:macros/latex/required/psnfss/psfonts.dtx](#)
- [3] Package `hyperref`, Sebastian Rahtz, Heiko Oberdiek, 2006/02/12.[CTAN:macros/latex/contrib/hyperref/](#)
- [4] Package `keyval`, David Carlisle, 1999/03/16.[CTAN:macros/latex/required/graphics/keyval.dtx](#)
- [5] Package `multicol`, Frank Mittelbach, 2004/02/14.[CTAN:macros/latex/required/tools/multicol.dtx](#)
- [6] Package `tabularx`, David Carlisle, 1999/01/07.[CTAN:macros/latex/required/tools/tabularx.dtx](#)
- [7] Package `tracefnt`, Frank Mittelbach, Rainer Schöpf, 1997/05/29.[CTAN:macros/latex/base/ltfsstrc.dtx](#)

- [8] Package `xkeyval`, Hendri Adriaens, 2005/05/07. [CTAN:macros/latex/contrib/xkeyval/](#)
- [9] The L^AT_EX3 Project, *L^AT_EX 2_ε 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 L^AT_EX’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 L^AT_EX. 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).

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; numbers in roman refer to the code lines where the entry is used.

Symbols	
<code>\#</code>	1145
<code>\%</code>	1148
<code>\@</code>	1116, 1141
<code>\@@fileswith@pti@ns</code>	827, 1105
<code>\@unprocessedoptions</code>	958, 993
<code>\@SetupDriver</code>	41, 43
<code>\@badrequireerror</code>	1106
<code>\@classoptionslist</code>	568, 570, 769, 787, 873, 875
<code>\@cls@pkg</code>	934, 969, 984, 987
<code>\@clsextension</code> .	248, 249, 268, 288, 289, 313, 326, 359, 387, 388, 409, 419, 440, 520, 532, 543, 566, 599, 766, 785, 843, 872, 975
<code>\@current</code>	199, 206, 209, 212, 217, 220, 223, 248, 268, 269, 288, 359, 387, 427, 566, 593, 597, 599, 624, 666, 750, 756, 766, 785, 843, 914, 915, 920, 921, 924, 929, 951, 953, 955, 957, 959, 960, 964, 971, 975, 994, 995, 999
<code>\@currname</code>	57, 196, 206, 209, 210, 212, 217, 220, 221, 223, 247, 269, 287, 364, 386, 427, 593, 597, 624, 666, 750, 756, 913, 915, 953, 955, 957, 959, 960, 984, 987, 995, 999
<code>\@curroptions</code>	751, 754, 771, 816, 996, 998, 1001
<code>\@declaredoptions</code>	762, 790, 823
<code>\@documentclasshook</code>	880
<code>\@ehc</code>	275, 455, 504, 540
<code>\@ehd</code>	708
<code>\@empty</code>	88, 89, 437, 486, 565, 607, 652, 660, 663, 673, 677, 749, 751, 763, 778, 788, 793, 826, 845, 903, 915, 916, 953, 996, 1002, 1010, 1030, 1031, 1044, 1087, 1092
<code>\@expandtwoargs</code>	585
<code>\@fileswith@pti@ns</code> ...	827, 1106, 1108
<code>\@firstofone</code>	308, 311, 379
<code>\@firstoftwo</code>	1022
<code>\@for</code>	570, 602, 631, 662, 762, 787, 803, 816, 823, 863, 1001, 1086
<code>\@gobble</code>	322, 482, 687
<code>\@gobbletwo</code>	377, 691
<code>\@if@pti@ns</code>	723
<code>\@if@ptions</code>	719, 921
<code>\@ifdefinable</code>	302
<code>\@ifl@aded</code>	920
<code>\@ifl@ter</code>	964
<code>\@ifnextchar</code>	340
<code>\@ifpackageloaded</code>	711
<code>\@ifstar</code>	554, 687, 759
<code>\@ifundefined</code>	178, 181, 186, 189, 195, 198, 209, 220, 235, 262, 450, 571, 593, 603, 624, 632, 735, 750, 817, 924, 995
<code>\@latex@error</code>	934, 982
<code>\@latex@warning@no@line</code>	966
<code>\@let@token</code>	1046, 1050
<code>\@missingfileerror</code>	957
<code>\@namedef</code>	348, 398, 427
<code>\@nameuse</code>	666
<code>\@one</code>	201, 1125, 1133
<code>\@nil</code> ..	572, 604, 633, 655, 665, 671, 678, 681, 893, 904, 1011, 1014, 1016, 1018, 1027, 1046, 1053, 1058, 1060, 1063, 1065, 1069, 1082
<code>\@onelevel@sanitize</code>	182, 306, 337, 338, 584, 606
<code>\@pass@ptions</code>	733, 951
<code>\@pkgextension</code>	251, 291, 390, 444, 891, 898, 994
<code>\@popfilename</code>	976
<code>\@pushfilename</code>	912
<code>\@removeelement</code>	585, 833
<code>\@reset@ptions</code>	917, 977
<code>\@secondoftwo</code>	1024
<code>\@tempc</code>	564
<code>\@twoloadclasserror</code>	975

<code>\@typeset@protect</code>	193	<code>\DeclareVoidOption</code> ..	5, 32, 33, 34, 375
<code>\@undefined</code>	961	<code>\default@ds</code>	819
<code>\@unknownoptionerror</code> ..	68, 981, 1002	<code>\define@key</code>	204, 215,
<code>\@unprocessedoptions</code>	653, 828, 958, 962		246, 286, 356, 385, 438, 442, 507
<code>\@unusedoptionlist</code> ..	586, 607, 608,	<code>\detokenize</code>	
	611, 613, 836, 839, 844, 845, 847		724, 728, 834, 836, 847, 849, 1021
<code>\@x@protect</code>	190	<code>\DisableKeyvalOption</code>	6, 446
<code>\[</code>	1146	<code>\do</code>	570, 602, 631, 662, 762,
<code>\]</code>	350, 1021, 1142		787, 803, 816, 823, 863, 1001, 1086
<code>\{</code>	1143	<code>\documentclass</code>	707, 944
<code>\}</code>	1144	<code>\ds@</code>	749
<code>\]</code>	1147		
		E	
<code>_</code>	1149	<code>\emph</code>	48, 87, 91, 93
		<code>\empty</code>	109
A		<code>\end</code>	1156
<code>\advance</code>	1125, 1133	<code>\endcsname</code>	105, 115, 136,
<code>\AtEndOfPackage</code>	653, 828		149, 152, 206, 212, 217, 223,
			233, 234, 242, 280, 281, 282,
C			283, 302, 332, 369, 394, 457,
<code>\catcode</code>	100, 101, 102, 103,		465, 469, 481, 485, 508, 513,
	104, 128, 129, 130, 131, 132,		514, 516, 517, 597, 685, 695,
	133, 134, 135, 153, 155, 159,		725, 737, 742, 744, 756, 778,
	161, 201, 202, 689, 1116, 1122,		793, 824, 840, 856, 864, 915,
	1131, 1141, 1142, 1143, 1144,		929, 953, 959, 960, 971, 999, 1155
	1145, 1146, 1147, 1148, 1149, 1150	<code>\endinput</code>	124, 692, 699, 709, 717
<code>\ClassError</code>	533	<code>\etex@unexpanded</code>	575, 579,
<code>\ClassInfo</code>	327, 360, 420, 521		1036, 1043, 1072, 1073, 1077, 1078
<code>\ClassWarning</code>	314, 410, 544	<code>\ExecuteOptions</code>	854
<code>\count@</code>	1118, 1122,		
	1124, 1125, 1129, 1131, 1132, 1133	F	
<code>\csname</code>	105, 115, 136,	<code>\futurelet</code>	1046
	149, 152, 205, 212, 216, 223,		
	233, 234, 242, 280, 281, 282,	G	
	283, 302, 332, 369, 394, 457,	<code>\gdef</code>	610, 736, 741, 875
	465, 469, 481, 485, 508, 513,		
	514, 516, 517, 597, 685, 695,	I	
	725, 736, 741, 744, 756, 778,	<code>\ifcase</code>	106
	793, 824, 840, 856, 864, 915,	<code>\ifetex@unexpanded</code>	702
	929, 953, 959, 960, 971, 999, 1155	<code>\ifin@</code>	776, 791, 809
<code>\CurrentOption</code>	35, 37, 38, 41,	<code>\ifKV@dyn@global</code>	463,
	58, 62, 64, 403, 600, 602, 604,		467, 474, 479, 483, 490, 511, 527
	606, 608, 614, 619, 629, 631,	<code>\ifMCS@print</code>	79
	633, 636, 640, 652, 662, 663,	<code>\ifnum</code>	1124, 1132
	665, 674, 678, 762, 763, 773,	<code>\ifpdf</code>	25
	778, 787, 788, 790, 793, 816,	<code>\ifx</code>	56, 88, 107, 109, 115, 136,
	823, 824, 826, 834, 849, 859,		144, 248, 268, 288, 307, 310,
	863, 864, 869, 916, 1001, 1002, 1007		313, 326, 350, 359, 387, 407,
<code>\CurrentOption@SaveLevel</code>			409, 419, 500, 520, 532, 543,
 853, 856, 860, 861, 866, 867		566, 568, 599, 607, 660, 663,
<code>\CurrentOptionKey</code>	674, 677		673, 685, 695, 763, 766, 785,
<code>\CurrentOptionValue</code>			788, 804, 843, 845, 872, 873,
 56, 672, 673, 675, 682		893, 895, 975, 994, 1002, 1021,
			1031, 1050, 1087, 1089, 1092, 1106
D		<code>\immediate</code>	117, 138
<code>\DeclareBoolOption</code> .	4, 20, 22, 229, 433	<code>\in@</code>	773
<code>\DeclareComplementaryOption</code>		<code>\in@false</code>	801
 5, 261, 434	<code>\in@true</code>	805, 810
<code>\DeclareDefaultOption</code>	6, 55, 426	<code>\input</code>	1113, 1151
<code>\DeclareOption</code>	201, 202	<code>\InputIfFileExists</code>	954
<code>\DeclareStringOption</code>			
 4, 26, 28, 50, 52, 339, 435	K	
		<code>\KV@@sp@def</code>	682

\KV@setcurrentvalue	681	\KV@dyn@action	450, 453, 457
\KV@action@error	493	\KV@dyn@ext	437, 440, 444, 509, 520
\KV@action@ignore	477	\KV@dyn@name	436, 439, 443, 500, 509, 525
\KV@action@undef	461		
\KV@action@warning	496	L	
\KV@AtEnd	157, 158, 692	\LoadClass	975
\KV@boolkey	247, 287, 304	\loop	1119, 1130
\KV@calldefault	648, 656		
\KV@checkfirstA	1055, 1058	M	
\KV@checkfirsttok	1046, 1049	\makeatletter	918
\KV@checkkv	1016, 1027	\MCS@driver	84
\KV@CurrentOption	570, 572, 580, 584, 585	\MCS@emph	88, 94
\KV@DeclareStringOption	341, 343, 346	\meaning	183, 453, 929, 933, 1007
\KV@disable@error	531	\MessageBreak	58, 237, 238, 264, 269, 271, 273, 274, 319, 415, 453, 473, 489, 502, 526, 538, 549, 705, 713, 714, 936, 937, 939, 940, 941, 943, 945, 968, 969, 970, 971, 987
\KV@disable@warning	542		
\KV@do@action	494, 497, 499	N	
\KV@ExecuteOptions	855, 858	\NeedsTeXFormat	4
\KV@false	310, 335	\newcommand	40, 43, 226, 229, 261, 339, 375, 426, 446, 553
\KV@family	204, 246, 270, 286, 356, 385, 557	\next	377, 379, 382
\KV@fileswith@ptions	871, 1105, 1108	\numexpr	861, 867
\KV@getkey	571, 604, 633, 655		
\KV@ifdefinable	230, 231, 232, 278, 279, 301, 347, 378	O	
\KV@ifempty	1015, 1018, 1020, 1028, 1059, 1062, 1070	\on@line	967
\KV@in@	790, 800	\OptionNotUsed	842
\KV@nil	1046, 1069, 1073, 1082		
\KV@normalize	721, 734, 874, 1009	P	
\KV@onefilewithoptions	879, 884, 890, 897, 911	\PackageError	263, 451, 501, 535, 704
\KV@param	305, 306, 307, 310, 319, 331, 332	\PackageInfo	120, 329, 362, 422, 472, 488, 523, 693
\KV@prefix	215, 230, 231, 232, 234, 235, 242, 253, 262, 272, 278, 279, 280, 281, 282, 283, 293, 347, 348, 369, 378, 394, 398	\PackageWarning	236, 316, 412, 546
\KV@process@ptions	782, 798, 815	\PackageWarningNoLine	57, 696, 712
\KV@process@ptions	759, 761	\PassOptionsToPackage	63, 80
\KV@ProcessKeyvalOptions	557, 561, 563	\ProcessKeyvalOptions	3, 74, 553, 686
\KV@removeelement	1083	\ProcessOptions	203, 748
\KV@removegarbage	1073, 1082	\protect	191
\KV@removelastspace	1052, 1060, 1063, 1065, 1069	\ProvidesPackage	5, 150
\KV@result	1010, 1012, 1035, 1036, 1042, 1043, 1071, 1072, 1076, 1077		
\KV@SanitizedCurrentOption	817, 840, 983, 986, 1006	R	
\KV@setcurrents	665, 671	\RangeCatcodeInvalid	1128, 1137, 1138, 1139, 1140
\KV@setcurrentvalue	678, 681	\renewcommand	93
\KV@splitcomma	1011, 1014, 1018	\repeat	1126, 1134
\KV@temp	565, 574, 576, 594, 601, 626, 630, 647, 651, 721, 728, 734, 738, 744, 874, 876, 889, 897, 906	\RequirePackage	7, 8, 84, 179, 701, 706
\KV@true	307, 335	\reserved@a	878, 883, 903, 904, 907, 909, 919, 979
\KV@use@ption	777, 792, 818, 821, 830	\reserved@b	892, 900, 904
\KV@VOID@	385, 401, 407	\RestoreCatcodes	1117, 1120, 1121, 1152
\KV@voidkey	386, 402		
\KV@x	1030, 1031, 1036	S	
\KV@xprocess@ptions	759, 784	\setkeys	44, 227, 448, 649
		\SetupDriver	32, 33, 34, 35, 38, 40
		\SetupKeyvalOptions	3, 13, 226, 429
		\space	268, 934, 937, 940, 942, 946, 969, 984, 987, 989, 990
		\strip@prefix	183, 187, 453, 928, 933, 1007

T		W	
\t	1091, 1092	\write	117, 138
\Test	1136, 1154		
\textcolor	94		
		X	
\the	153, 159, 356, 365, 600, 619, 626, 629, 636, 640, 648, 649, 773, 861, 867, 906, 1091, 1094, 1096, 1101, 1122	\x	105, 107, 109, 116, 120, 122, 137, 142, 149, 245, 256, 285, 296, 355, 372, 384, 397, 405, 407, 449, 459, 462, 478, 506, 556, 559, 658, 660, 722, 731, 772, 775, 803, 804, 808, 813, 832, 839, 925, 927, 937, 943, 1086, 1087, 1089, 1096, 1100, 1103
\TMP@EnsureCode	156, 163, 164, 165, 166, 167, 168, 169, 170, 171, 172, 173, 174, 175, 176, 177		
\toks	592, 635, 636, 648, 771, 773		
\toks@	351, 353, 356, 358, 365, 594, 596, 600, 601, 618, 619, 625, 626, 629, 630, 639, 640, 649, 767, 769, 773, 905, 906, 1085, 1091, 1094, 1096, 1101		
\tw@	202, 592, 635, 636, 648, 689, 771, 773		
		Y	
		\y	932, 933, 940, 943
U		Z	
\unexpanded	705	\zap@space	677, 903, 1030, 1044