

# The kvoptions package

Heiko Oberdiek  
<oberdiek@uni-freiburg.de>

2006/08/22 v2.4

## Abstract

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

## Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
1.1	The beginning . . . . .	2
1.2	Overview . . . . .	2
<b>2</b>	<b>Usage</b>	<b>3</b>
2.1	Process options . . . . .	3
2.1.1	\ProcessKeyvalOptions . . . . .	3
2.1.2	\SetupKeyvalOptions . . . . .	3
2.2	Option declarations . . . . .	3
2.2.1	\DeclareStringOption . . . . .	4
2.2.2	\DeclareBoolOption . . . . .	4
2.2.3	\DeclareComplementaryOption . . . . .	5
2.2.4	\DeclareVoidOption . . . . .	5
2.2.5	\DeclareDefaultOption . . . . .	5
2.2.6	Dynamic options . . . . .	5
2.2.7	\DisableKeyvalOption . . . . .	6
2.3	Summary of internal macros . . . . .	7
<b>3</b>	<b>Example</b>	<b>7</b>
<b>4</b>	<b>Package options</b>	<b>9</b>
4.1	Option patch . . . . .	9
4.2	Option debugshow . . . . .	10
<b>5</b>	<b>Limitations</b>	<b>10</b>
5.1	Compatibility . . . . .	10
5.1.1	Option patch vs. package xkvtxp . . . . .	10
5.2	Limitations . . . . .	11
5.2.1	Option comparisons . . . . .	11
5.2.2	Option list parsing with option patch . . . . .	11
<b>6</b>	<b>Implementation</b>	<b>11</b>
6.1	Preamble . . . . .	11
6.2	Option declaration macros . . . . .	12
6.2.1	\SetupKeyvalOptions . . . . .	12
6.2.2	\DeclareBoolOption . . . . .	13
6.2.3	\DeclareStringOption . . . . .	15
6.2.4	\DeclareVoidOption . . . . .	16
6.2.5	\DeclareDefaultOption . . . . .	17

6.3	Dynamic options	17
6.3.1	<code>\DisableKeyvalOption</code>	17
6.4	Process options	19
6.5	<code>\ProcessKeyvalOptions</code>	19
6.5.1	Helper macros	21
6.6	Patch	21
<b>7</b>	<b>Installation</b>	<b>28</b>
7.1	Some details for the interested	29
<b>8</b>	<b>References</b>	<b>29</b>
<b>9</b>	<b>History</b>	<b>30</b>
[0000/00/00 v0.0]		30
[2004/02/22 v1.0]		30
[2006/02/16 v2.0]		30
[2006/02/20 v2.1]		30
[2006/06/01 v2.2]		30
[2006/08/17 v2.3]		30
[2006/08/22 v2.4]		30
<b>10</b>	<b>Index</b>	<b>31</b>

# 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  $\text{\LaTeX}$ 's package and class `options`:

Package <code>keyval</code>	Package <code>kvoptions</code>	$\text{\LaTeX}$ 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<sup>A</sup>T<sub>E</sub>X'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.

Note, neither of the following commands 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<sup>A</sup>T<sub>E</sub>X 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>	<code>(foobar)</code>	<code>\ProcessKeyvalOptions*</code> <code>\DeclareBoolOption</code> <code>\DeclareStringOption</code>
prefix	<code>⟨\currname⟩@</code>	<code>(foobar@)</code>	<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:

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

Then `\DeclareStringOption` defines the macro with content `me`, note L<sup>A</sup>T<sub>E</sub>X 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\iffobar@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 `final` 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 2.1.2). 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 key, it is the equivalent to L<sup>A</sup>T<sub>E</sub>X's `\DeclareOption*`. It allows the specification of a default action `<code>` that is invoked if an unknown option is found. During the execution of `<code>` `\CurrentOption` contains the current option string. Additionally `\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<sup>A</sup>T<sub>E</sub>X 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<sup>A</sup>T<sub>E</sub>X 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<sup>A</sup>T<sub>E</sub>X'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` 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>:
  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 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 $\langle key \rangle$	Defined macro	Description
<code>\DeclareStringOption</code>	$\langle prefix \rangle \langle key \rangle$	holds the string
<code>\DeclareBoolOption</code>	$\langle prefix \rangle \langle key \rangle$ $\langle prefix \rangle \langle key \rangle false$ $\langle prefix \rangle \langle key \rangle true$	boolean switch disable switch enable switch
<code>\DeclareComplementaryOption</code>	$\langle prefix \rangle \langle key \rangle false$ $\langle prefix \rangle \langle key \rangle true$	enable parent switch disable parent switch
<code>\DeclareVoidOption</code>	$\langle prefix \rangle \langle key \rangle$	holds the action

## 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}[2006/08/22 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\expandafter{\CurrentOption}{color}%
64   \else
65     % Package color does not take options with values.
66     % We provide the standard LaTeX error.
67     \@unknownoptionerror
68   \fi
69 }
70
71 % Process options
72 % -----
73 \ProcessKeyvalOptions*
74
75 % Implementation depending on option values
76 % -----
77 % Code for print mode
78 \ifMCS@print
79   \PassOptionsToPackage{monochrome}{color}
80   % tells package color to use black and white
81 \fi
82
83 \RequirePackage[\MCS@driver]{color}
84 % load package color with the correct driver
85
86 % \emph setup
87 \ifx\MCS@emph\@empty
88   % \@empty is a predefined macro with empty contents.
89   % the option value of option emph is empty, thus
90   % we do not want a redefinition of \emph.
91 \else
92   \renewcommand*{\emph}[1]{%

```



```

93     \textcolor{\MCS@emph}{#1}%
94   }
95 \fi
96 \end{example}

```

## 4 Package options

The package `koptions` knows two package options `patch` and `debugshow`. The options of package `koptions` 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  $\text{\LaTeX}$  is unhappy if a package is loaded later again with options that are previously not given. Thus package and class authors, stay with `\RequirePackage{koptions}` without options.

### 4.1 Option `patch`

$\text{\LaTeX}$ '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.  $\text{\LaTeX}$  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,  $\text{\LaTeX}$  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  $\text{\LaTeX}$ '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 and 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  $\text{\LaTeX}$ '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 L<sup>A</sup>T<sub>E</sub>X compares options, e.g. option clash check, `\@ifpackagewith`, `\@ifclasswith`. Apart from catcode and sanitizing problems of option `patch`, there is another problem. L<sup>A</sup>T<sub>E</sub>X 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

**Package identification.**

```

97 <*package>
98 \NeedsTeXFormat{LaTeX2e}
99 \ProvidesPackage{kvoptions}
100 [2006/08/22 v2.4 Connects package keyval with LaTeX options (H0)]

```

**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`.

```
101 \RequirePackage{keyval}
```

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

```
102 \begingroup
103   \edef\x{\endgroup
104     \noexpand\AtEndOfPackage{%
105       \catcode\@ne=\the\catcode\@ne\relax
106       \catcode\tw=\the\catcode\tw\relax
107     }%
108   }%
109 \x
110 \catcode\@ne=14 %
111 \catcode\tw=14 %
112 \DeclareOption{debugshow}{\catcode\@ne=9 }
113 \DeclareOption{patch}{\catcode\tw=9 }
    Optionen auswerten:
114 \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.

`\KV@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<sup>A</sup>T<sub>E</sub>X's class/package system.

```
115 \define@key{KV@}{family}{%
116   \expandafter\edef\csname KV@family@\currname.\@current\endcsname{#1}%
117 }
118 \def\KV@family{%
119   \ifundefined{KV@family@\currname.\@current}{%
120     \currname
121   }{%
122     \csname KV@family@\currname.\@current\endcsname
123   }%
124 }
```

`\KV@prefix` The value settings of options, 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.

```
125 \define@key{KV@}{prefix}{%
126   \expandafter\edef\csname KV@prefix@\currname.\@current\endcsname{#1}%
127 }
128 \def\KV@prefix{%
129   \ifundefined{KV@prefix@\currname.\@current}{%
130     \currname @%
131   }{%
132     \csname KV@prefix@\currname.\@current\endcsname
133   }%
134 }
```

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

```

135 \newcommand*{\SetupKeyvalOptions}{%
136   \setkeys{KV0}%
137 }
```

## 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<sup>A</sup>T<sub>E</sub>X's `\newif` does not check for already defined macros, therefore we add this check here to prevent the user from accidentally redefining of T<sub>E</sub>X's primitives and other macros.

```

138 \newcommand*{\DeclareBoolOption}[2][false]{%
139   \KV0@ifdefinable{if\KV0@prefix#2}{%
140     \KV0@ifdefinable{\KV0@prefix#2true}{%
141       \KV0@ifdefinable{\KV0@prefix#2false}{%
142         \expandafter\newif\csname if\KV0@prefix#2\endcsname
143         \@ifundefined{\KV0@prefix#2#1}{%
144           \PackageWarning{kvoptions}{%
145             Initialization of option ‘#2’ failed,\MessageBreak
146             cannot set boolean option to ‘#1’,\MessageBreak
147             use ‘true’ or ‘false’, now using ‘false’%
148           }%
149         }{%
150           \csname\KV0@prefix#2#1\endcsname
151         }%
152         \begingroup
153         \edef\x{\endgroup
154           \noexpand\define@key{\KV0@family}{#2}[true]{%
155             \noexpand\KV0@boolkey{\@currname}%
156             \ifx\@currentx\@clsextension
157               \noexpand\@clsextension
158             \else
159               \noexpand\@pkgextension
160             \fi
161             {\KV0@prefix}{#2}{####1}%
162           }%
163         }%
164         \x
165       }%
166     }%
167   }%
168 }
```

`\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.

```

169 \newcommand*{\DeclareComplementaryOption}[2]{%
170   \@ifundefined{if\KV0@prefix#2}{%
171     \PackageError{kvoptions}{%
172       Cannot generate option code for ‘#1’,\MessageBreak
173       parent switch ‘#2’ does not exist%
174     }{%
175       You are inside %
176       \ifx\@currentx\@clsextension class\else package\fi\space
177       ‘\@currname.\@currentx’.\MessageBreak
178     }%
179   }
```

```

178     '\KV0@family' is used as familiy for the keyval options.\MessageBreak
179     '\KV0@prefix' serves as prefix for internal switch macros.\MessageBreak
180     \MessageBreak
181     \@ehc
182 }%
183 }{%
184     \KV0@ifdefinable{\KV0@prefix#1true}{%
185     \KV0@ifdefinable{\KV0@prefix#1false}{%
186         \expandafter\let\csname\KV0@prefix#1false\expandafter\endcsname
187         \csname\KV0@prefix#2true\endcsname
188     \expandafter\let\csname\KV0@prefix#1true\expandafter\endcsname
189     \csname\KV0@prefix#2false\endcsname

```

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

```

190     \begingroup
191     \edef\x{\endgroup
192         \noexpand\define@key{\KV0@family}{#1}[true]{%
193             \noexpand\KV0@boolkey{\@currname}%
194             \ifx\@currentx\@clsextension
195                 \noexpand\@clsextension
196             \else
197                 \noexpand\@pkgextension
198             \fi
199             {\KV0@prefix}{#1}{####1}%
200         }%
201     }%
202     \x
203 }%
204 }%
205 }%
206 }

```

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

```

207 \def\KV0@ifdefinable#1{%
208     \expandafter\@ifdefinable\csname #1\endcsname
209 }

```

`\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

210 \def\KV0@boolkey#1#2#3#4#5{%
211     \edef\KV0@param{#5}%
212     \@onelevel@sanitize\KV0@param
213     \ifx\KV0@param\KV0@true
214         \expandafter\@firstofone
215     \else
216         \ifx\KV0@param\KV0@false
217             \expandafter\expandafter\expandafter\@firstofone
218         \else
219             \ifx#2\@clsextension
220                 \expandafter\ClassWarning
221             \else
222                 \expandafter\PackageWarning
223             \fi
224             {#1}{%
225                 Value '\KV0@param' is not supported by\MessageBreak
226                 option '#4'%

```

```

227     }%
228     \expandafter\expandafter\expandafter\@gobble
229   \fi
230 \fi
231 {%
232   ^^A\ifx#2\@clsextension
233   ^^A \expandafter\ClassInfo
234   ^^A\else
235   ^^A \expandafter\PackageInfo
236   ^^A\fi
237   ^^A{#1}{[option] #4=\KV0@param}%
238   \csname#3#4\KV0@param\endcsname
239 }%
240 }

```

`\KV0@true` The macros `\KV0@true` and `\KV0@false` are used for string comparisons. After `\KV0@false` `\@onelevel@sanitize` we have only tokens with catcode 12 (other).

```

241 \def\KV0@true{true}
242 \def\KV0@false{false}
243 \@onelevel@sanitize\KV0@true
244 \@onelevel@sanitize\KV0@false

```

### 6.2.3 \DeclareStringOption

`\DeclareStringOption`

```

245 \newcommand*{\DeclareStringOption}[2][]{%
246   \@ifnextchar[%
247     \KV0@DeclareStringOption{#1}{#2}%
248   }{%
249     \KV0@DeclareStringOption{#1}{#2}{}%
250   }%
251 }

```

`\KV0@DeclareStringOption`

```

252 \def\KV0@DeclareStringOption#1#2#3[#4]{%
253   \KV0@ifdefinable{\KV0@prefix#2}{%
254     \@namedef{\KV0@prefix#2}{#1}%
255     \begingroup
256       \ifx\#3\%
257         \toks@{%
258           \else
259             \toks@{[#{#4}]}%
260           \fi
261         \edef\x{\endgroup
262           \noexpand\define@key{\KV0@family}{#2}\the\toks@{%
263             ^^A\begingroup
264             ^^A \toks@{####1}%
265             ^^A \ifx\@current\@clsextension
266             ^^A \noexpand\ClassInfo
267             ^^A \else
268             ^^A \noexpand\PackageInfo
269             ^^A \fi
270             ^^A {\@currname}{%
271               [option] #2={\noexpand\the\toks@}%
272             }%
273             ^^A\endgroup
274             \noexpand\def
275             \expandafter\noexpand\csname\KV0@prefix#2\endcsname{####1}%
276           }%
277         }%
278       \x
279     }%

```

280 }

#### 6.2.4 \DeclareVoidOption

\DeclareVoidOption

```
281 \newcommand*{\DeclareVoidOption}[1]{%
282   \begingroup
283     \let\next\@gobbletwo
284     \KV@ifdefinable{\KV@prefix#1}{%
285       \let\next\@firstofone
286     }%
287   \expandafter\endgroup
288   \next{%
289     \begingroup
290     \edef\x{\endgroup
291       \noexpand\define@key{\KV@family}{#1}[\KV@VOID@]{%
292         \noexpand\KV@voidkey{\@currname}%
293         \ifx\@current\@clsextension
294           \noexpand\@clsextension
295         \else
296           \noexpand\@pkgextension
297         \fi
298         {#1}%
299         {###1}%
300       \expandafter\noexpand\csname\KV@prefix#1\endcsname
301     }%
302   }%
303   \x
304   \@namedef{\KV@prefix#1}%
305 }%
306 }
307 \def\KV@VOID@{@VOID@}
```

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

\KV@voidkey

```
308 \def\KV@voidkey#1#2#3#4{%
309   \def\CurrentOption{#3}%
310   \begingroup
311     \def\x{#4}%
312   \expandafter\endgroup
313   \ifx\x\KV@VOID@
314   \else
315     \ifx#2\@clsextension
316       \expandafter\ClassWarning
317     \else
318       \expandafter\PackageWarning
319     \fi
320     {#1}{%
321       Unexpected value for option ‘#3’\MessageBreak
322       is ignored%
323     }%
324   \fi
325   ^^A\ifx#2\@clsextension
326   ^^A \expandafter\ClassInfo
327   ^^A\else
328   ^^A \expandafter\PackageInfo
329   ^^A\fi
330   ^^A{#1}{[option] #3}%
331 }
```



### 6.2.5 \DeclareDefaultOption

\DeclareDefaultOption

```
332 \newcommand*{\DeclareDefaultOption}{%
333   \namedef{KV0@default@ \@currname.\@current}{%
334   }
```

## 6.3 Dynamic options

### 6.3.1 \DisableKeyvalOption

```
335 \SetupKeyvalOptions{%
336   family=KV0dyn,%
337   prefix=KV0dyn%
338 }
339 \DeclareBoolOption[true]{global}
340 \DeclareComplementaryOption{local}{global}
341 \DeclareStringOption[undef]{action}
342 \let\KV0dyn@name\relax
343 \let\KV0dyn@ext\@empty
344 \define@key{KV0dyn}{class}{%
345   \def\KV0dyn@name{#1}%
346   \let\KV0dyn@ext\@clsextension
347 }
348 \define@key{KV0dyn}{package}{%
349   \def\KV0dyn@name{#1}%
350   \let\KV0dyn@ext\@pkgextension
351 }
352 \newcommand*{\DisableKeyvalOption}[3][[]]{%
353   \begingroup
354     \setkeys{KV0dyn}{#1}%
355     \def\x{\endgroup}%
356     \ifundefined{KV0@action@KV0dyn@action}{%
357       \PackageError{kvoptions}{%
358         Unknown disable action
359         '\expandafter\strip@prefix\meaning\KV0dyn@action'\MessageBreak
360         for option '#3' in keyval family '#2'%
361       }\@ehc
362     }{%
363       \csname KV0@action@KV0dyn@action\endcsname{#2}{#3}%
364     }%
365   \x
366 }
367 \def\KV0@action@undef#1#2{%
368   \edef\x{\endgroup
369     \ifKV0dyn@global\global\fi
370     \let
371     \expandafter\noexpand\csname KV@#1@#2\endcsname
372     \relax
373     \ifKV0dyn@global\global\fi
374     \let
375     \expandafter\noexpand\csname KV@#1@#2@default\endcsname
376     \relax
377   }%
378   ^^A\PackageInfo{kvoptions}{%
379     ^^A [option] key '#2' of family '#1'\MessageBreak
380     ^^A is disabled (undef, \ifKV0dyn@global\global\else local\fi)%
381   ^^A}%
382 }
383 \def\KV0@action@ignore#1#2{%
384   \edef\x{\endgroup
385     \ifKV0dyn@global\global\fi
386     \let
```

```

387 \expandafter\noexpand\csname KV@#1@#2\endcsname
388 \@gobble
389 \ifKV0dyn@global\global\fi
390 \let
391 \expandafter\noexpand\csname KV@#1@#2@default\endcsname
392 \@empty
393 }%
394 ^^A\PackageInfo{kvoptions}{%
395 ^^A [option] key '#2' of family '#1'\MessageBreak
396 ^^A is disabled (ignore, \ifKV0dyn@global\global\else local\fi)%
397 ^^A}%
398 }
399 \def\KV0@action@error{%
400 \KV0@do@action{error}%
401 }
402 \def\KV0@action@warning{%
403 \KV0@do@action{warning}%
404 }

#1 error or warning
#2 <family>
#3 <key>
405 \def\KV0@do@action#1#2#3{%
406 \ifx\KV0dyn@name\relax
407 \PackageError{kvoptions}{%
408 Action type '#1' needs package/class name\MessageBreak
409 for key '#3' in family '#2'%
410 }\@ehc
411 \else
412 \edef\x{\endgroup
413 \noexpand\define@key{#2}{#3}[]{%
414 \expandafter\noexpand\csname KV0@disable@#1\endcsname
415 {\KV0dyn@name}\noexpand\KV0dyn@ext{#3}%
416 }%
417 \ifKV0dyn@global
418 \global\let
419 \expandafter\noexpand\csname KV@#2@#3\endcsname
420 \expandafter\noexpand\csname KV@#2@#3\endcsname
421 \global\let
422 \expandafter\noexpand\csname KV@#2@#3@default\endcsname
423 \expandafter\noexpand\csname KV@#2@#3@default\endcsname
424 \fi
425 }%
426 ^^A\ifx\KV0dyn@ext\@clsextension
427 ^^A \expandafter\ClassInfo
428 ^^A\else
429 ^^A \expandafter\PackageInfo
430 ^^A\fi
431 ^^A{\KV0dyn@name}{%
432 ^^A [option] key '#3' of family '#2'\MessageBreak
433 ^^A is disabled (#1, \ifKV0dyn@global\global\else local\fi)%
434 ^^A}%
435 \fi
436 }
437 \def\KV0@disable@error#1#2#3{%
438 \ifx#2\@clsextension
439 \expandafter\ClassError
440 \else
441 \expandafter\PackageError
442 \fi
443 {#1}{%
444 Option '#3' is given too late,\MessageBreak
445 now the option is ignored%

```

```

446 } \@ehc
447 }
448 \def\KV0@disable@warning#1#2#3{%
449   \ifx#2\@clsextension
450     \expandafter\ClassWarning
451   \else
452     \expandafter\PackageWarning
453   \fi
454   {#1}{%
455     Option ‘#3’ is already consumed\MessageBreak
456     and has no effect%
457   }%
458 }

```

## 6.4 Process options

### 6.5 \ProcessKeyvalOptions

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

```

459 \newcommand*{\ProcessKeyvalOptions}{%
460   \@ifstar{%
461     \begingroup
462     \edef\x{\endgroup
463       \noexpand\KV0@ProcessKeyvalOptions{\KV0@family}%
464     }%
465     \x
466   }%
467   \KV0@ProcessKeyvalOptions
468 }

```

```

469 \def\KV0@ProcessKeyvalOptions#1{%
470   \let\@tempc\relax
471   \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).

```

472   \ifx\@currentx\@clsextension
473   \else
474     \ifx\@classoptionslist\relax
475     \else
476       \@for\KV0@CurrentOption:=\@classoptionslist\do{%
477         \ifundefined{KV@#1}\expandafter\KV0@getkey\KV0@CurrentOption=\@nil}{%
478         }{%
479           \edef\KV0@temp{%
480             \unexpanded\expandafter{%
481               \KV0@temp
482             }%
483             ,%
484             \unexpanded\expandafter{%
485               \KV0@CurrentOption
486             }%
487             ,%
488           }%
489             \@onelevel@sanitize\KV0@CurrentOption
490             \@expandtwoargs\@removeelement\KV0@CurrentOption
491             \@unusedoptionlist\@unusedoptionlist
492           }%
493         }%
494       \fi
495     \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`.

```

496 \begingroup
497   \toks\tw@{}%
498   \@ifundefined{opt@\@currname.\@currentx}{%
499     \toks@\expandafter{\KVO@temp}%
500   }{%
501     \toks@\expandafter\expandafter\expandafter{%
502       \csname opt@\@currname.\@currentx\endcsname
503     }%
504     \ifx\@currentx\@clsextension
505       \edef\CurrentOption{\the\toks@}%
506       \toks@\expandafter{\KVO@temp}%
507       \@for\CurrentOption:=\CurrentOption\do{%
508         \@ifundefined{%
509           KV@#1@\expandafter\KVO@getkey\CurrentOption=\@nil
510         }{%

```

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

```

511 ^^B      \@onelevel@sanitize\CurrentOption
512          \ifx\@unusedoptionlist\@empty
513            \global\let\@unusedoptionlist\CurrentOption
514          \else
515            \expandafter\expandafter\expandafter\gdef
516            \expandafter\expandafter\expandafter\@unusedoptionlist
517            \expandafter\expandafter\expandafter{%
518              \expandafter\@unusedoptionlist
519              \expandafter,\CurrentOption
520            }%
521          \fi
522        }{%
523          \toks@\expandafter{%
524            \the\expandafter\toks@\expandafter,\CurrentOption
525          }%
526        }%
527      }%
528    \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.

```

529      \@ifundefined{KVO@default@\@currname.\@currentx}{%
530        \toks@\expandafter\expandafter\expandafter{%
531          \expandafter\KVO@temp\the\toks@
532        }%
533      }{%
534        \edef\CurrentOption{\the\toks@}%
535        \toks@\expandafter{\KVO@temp}%
536        \@for\CurrentOption:=\CurrentOption\do{%
537          \@ifundefined{%
538            KV@#1@\expandafter\KVO@getkey\CurrentOption=\@nil
539          }{%
540            \toks\tw@\expandafter{%
541              \the\toks@\expandafter\tw@\expandafter,\CurrentOption
542            }%
543          }{%
544            \toks@\expandafter{%
545              \the\expandafter\toks@\expandafter,\CurrentOption
546            }%
547          }%
548        }%
549      }%
550    \fi
551  }%
552 \edef\KVO@temp{\endgroup

```

```

553     \noexpand\KV0@calldefault{\the\toks\tw}%
554     \noexpand\setkeys{#1}{\the\toks@}%
555   }%
556   \KV0@temp
Some cleanup of \ProcessOptions.
557   \let\CurrentOption\@empty
558   \AtEndOfPackage{\let\@unprocessedoptions\relax}%
559 }

```

### 6.5.1 Helper macros

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

```

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

```

`\KV0@calldefault`

```

561 \def\KV0@calldefault#1{%
562   \begingroup
563     \def\x{#1}%
564   \expandafter\endgroup
565   \ifx\x\@empty
566   \else
567     \@for\CurrentOption:=#1\do{%
568       \ifx\CurrentOption\@empty
569       \else
570         \expandafter\KV0@setcurrents\CurrentOption=\@nil
571         \@nameuse{KV0@default@\currname.\@current}%
572       \fi
573     }%
574   \fi
575 }

```

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

```

576 \def\KV0@setcurrents#1=#2\@nil{
577   \def\CurrentOptionValue{#2}%
578   \ifx\CurrentOptionValue\@empty
579     \let\CurrentOptionKey\CurrentOption
580     \let\CurrentOptionValue\relax
581   \else
582     \edef\CurrentOptionKey{\zap@space#1 \@empty}%
583     \expandafter\KV0@setcurrentvalue\CurrentOption\@nil
584   \fi
585 }

```

`\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.

```

586 \def\KV0@setcurrentvalue#1=#2\@nil{%
587   \KV@@sp@def\CurrentOptionValue{#2}%
588 }

```

## 6.6 Patch

Caution: `docstrip` stops at `\endinput` at begin of line!

```

589 ^^B\@gobble
590   \endinput
591 \PackageInfo{kvoptions}{Patching LaTeX's option system}
Check for  $\varepsilon$ -TeX.
592 \begingroup\expandafter\expandafter\expandafter\endgroup
593 \expandafter\ifx\csname eTeXversion\endcsname\relax
594   \PackageWarningNoLine{kvoptions}{%
595     Option 'patch' ignored, because e-TeX is missing%

```

```

596 }%
597 \expandafter\endinput
598 \fi

    Check for package xkvltxp.
599 \@ifpackageloaded{xkvltxp}{%
600   \PackageWarningNoLine{kvoptions}{%
601     Option 'patch' cannot be used together with\MessageBreak
602     package 'xkvltxp' that is already loaded.\MessageBreak
603     Therefore option 'patch' will be ignored%
604   }%
605   \endinput
606 }{}

607 \def\@if@ptions#1#2#3{%
608   \begingroup
609   \KV0@normalize\KV0@temp{#3}%
610   \edef\x{\endgroup
611     \noexpand\@if@pti@ns{%
612       \detokenize\expandafter\expandafter\expandafter{%
613         \csname opt@#2.#1\endcsname
614       }%
615     }{%
616       \detokenize\expandafter{\KV0@temp}%
617     }%
618   }%
619   \x
620 }

621 \def\@pass@ptions#1#2#3{%
622   \KV0@normalize\KV0@temp{#2}%
623   \@ifundefined{opt@#3.#1}{%
624     \expandafter\gdef\csname opt@#3.#1%
625       \expandafter\endcsname\expandafter{%
626         \KV0@temp
627       }%
628   }{%
629     \expandafter\gdef\csname opt@#3.#1%
630       \expandafter\expandafter\expandafter\endcsname
631       \expandafter\expandafter\expandafter{%
632         \csname opt@#3.#1\expandafter\endcsname\expandafter,\KV0@temp
633       }%
634   }%
635 }

636 \def\ProcessOptions{%
637   \let\ds@\@empty
638   \@ifundefined{opt@\@currname.\@currentx}{%
639     \let\@curroptions\@empty
640   }{%
641     \expandafter\expandafter\expandafter\def
642     \expandafter\expandafter\expandafter\@curroptions
643     \expandafter\expandafter\expandafter{%
644       \csname opt@\@currname.\@currentx\endcsname
645     }%
646   }%
647   \@ifstar\KV0@xprocess@ptions\KV0@process@ptions
648 }

649 \def\KV0@process@ptions{%
650   \@for\CurrentOption:=\@declaredoptions\do{%
651     \ifx\CurrentOption\@empty
652       \else
653         \begingroup
654         \ifx\@currentx\@clsextension

```

```

655         \toks@{ }%
656     \else
657         \toks@expandafter{\@classoptionslist,}%
658     \fi
659     \toks\tw@expandafter{\@curroptions}%
660     \edef\x{\endgroup
661         \noexpand\in@{,\CurrentOption,}{,\the\toks@the\toks\tw@,}%
662     }%
663     \x
664     \ifin@
665         \KVO@use@option
666         \expandafter\let\csname ds@\CurrentOption\endcsname\@empty
667     \fi
668 \fi
669 }%
670 \KVO@process@ptions
671 }

672 \def\KVO@xprocess@ptions{%
673     \ifx\@current\@clsextension
674     \else
675         \@for\CurrentOption:=\@classoptionslist\do{%
676             \ifx\CurrentOption\@empty
677             \else
678                 \KVO@in@\CurrentOption\@declaredoptions
679                 \ifin@
680                     \KVO@use@option
681                     \expandafter\let\csname ds@\CurrentOption\endcsname\@empty
682                 \fi
683             \fi
684         }%
685     \fi
686     \KVO@process@ptions
687 }

688 \def\KVO@in@#1#2{%
689     \in@false
690     \begingroup
691         \@for\x:=#2\do{%
692             \ifx\x#1\relax
693                 \in@true
694             \fi
695         }%
696     \edef\x{\endgroup
697         \ifin@
698             \noexpand\in@true
699         \fi
700     }%
701     \x
702 }

703 \def\KVO@process@ptions{%
704     \@for\CurrentOption:=\@curroptions\do{%
705         \ifundefined{ds@\KVO@SanitizedCurrentOption}{%
706             \KVO@use@option
707             \default@ds
708         }%
709         \KVO@use@option
710     }%
711     \@for\CurrentOption:=\@declaredoptions\do{%
712         \expandafter\let\csname ds@\CurrentOption\endcsname\relax
713     }%
714     \let\CurrentOption\@empty
715     \let\@fileswith@ptions\@fileswith@ptions
716     \AtEndOfPackage{\let\@unprocessedoptions\relax}%

```

```

717 }
718 \def\KV0@use@option{%
719   \begingroup
720   \edef\x{\endgroup
721     \noexpand\removeelement{%
722       \detokenize\expandafter{\CurrentOption}%
723     }{%
724       \detokenize\expandafter{\@unusedoptionlist}%
725     }%
726   }%
727   \x\@unusedoptionlist
728   \csname ds@\KV0@SanitizedCurrentOption\endcsname
729 }
730 \def\OptionNotUsed{%
731   \ifx\@currentx\@clsextension
732     \xdef\@unusedoptionlist{%
733       \ifx\@unusedoptionlist\@empty
734         \else
735           \detokenize\expandafter{\@unusedoptionlist,}%
736         \fi
737       \detokenize\expandafter{\CurrentOption}%
738     }%
739   \fi
740 }

```

Variant of \ExecuteOptions that better protects \CurrentOption.

```

741 \def\CurrentOption@SaveLevel{0}
742 \def\ExecuteOptions{%
743   \expandafter\KV0@ExecuteOptions
744   \csname CurrentOption@\CurrentOption@SaveLevel\endcsname
745 }
746 \def\KV0@ExecuteOptions#1#2{%
747   \let#1\CurrentOption
748   \edef\CurrentOption@SaveLevel{\the\numexpr\CurrentOption@SaveLevel+1}%
749   \@for\CurrentOption:=#2\do{%
750     \csname ds@\CurrentOption\endcsname
751   }%
752   \edef\CurrentOption@SaveLevel{\the\numexpr\CurrentOption@SaveLevel-1}%
753   \let\CurrentOption#1%
754 }
755 \def\KV0@fileswith@pti@ns#1[#2]#3[#4]{%
756   \ifx#1\@clsextension
757     \ifx\@classoptionslist\relax
758       \KV0@normalize\KV0@temp{#2}%
759       \expandafter\gdef\expandafter\@classoptionslist\expandafter{%
760         \KV0@temp
761       }%
762       \def\reserved@a{%
763         \KV0@onefilewithoptions#3[#2][#4]#1%
764         \@documentclasshook
765       }%
766     \else
767       \def\reserved@a{%
768         \KV0@onefilewithoptions#3[#2][#4]#1%
769       }%
770     \fi
771   \else
772     \begingroup
773     \let\KV0@temp\relax
774     \let\KV0@onefilewithoptions\relax
775     \let\@pkgextension\relax
776     \def\reserved@b##1,{%

```



```

777     \ifx\@nil##1\relax
778     \else
779     \ifx\relax##1\relax
780     \else
781     \KV@onefilewithoptions##1[\KV@temp][#4]\@pkgextension
782     \fi
783     \expandafter\reserved@b
784 \fi
785 }%
786 \edef\reserved@a{\zap@space#3 \@empty}%
787 \edef\reserved@a{\expandafter\reserved@b\reserved@a,\@nil,}%
788 \toks@{#2}%
789 \def\KV@temp{\the\toks@}%
790 \edef\reserved@a{\endgroup \reserved@a}%
791 \fi
792 \reserved@a
793 }

794 \def\KV@onefilewithoptions#1[#2][#3]#4{%
795   \@pushfilename
796   \xdef\@currname{#1}%
797   \global\let\@currentx#4%
798   \expandafter\let\csname\@currname.\@currentx-h@k\endcsname\@empty
799   \let\CurrentOption\@empty
800   \@reset@options
801   \makeatletter
802   \def\reserved@a{%
803     \ifl@aded\@currentx{#1}{%
804       \@if@options\@currentx{#1}{#2}{%
805         }{%
806           \begingroup
807             \ifundefined{opt@#1.\@currentx}{%
808               \def\x{%
809                 }{%
810                   \edef\x{%
811                     \expandafter\expandafter\expandafter\strip@prefix
812                     \expandafter\meaning\csname opt@#1.\@currentx\endcsname
813                   }%
814                 }%
815               \def\y{#2}%
816               \edef\y{\expandafter\strip@prefix\meaning\y}%
817               \@latex@error{Option clash for \cls@pkg\space #1}{%
818                 The package #1 has already been loaded
819                 with options:\MessageBreak
820                 \space\space[\x]\MessageBreak
821                 There has now been an attempt to load it
822                 with options:\MessageBreak
823                 \space\space[\y]\MessageBreak
824                 Adding the global options:\MessageBreak
825                 \space\space
826                 \x,\y\MessageBreak
827                 to your \noexpand\documentclass declaration may fix this.%
828                 \MessageBreak
829                 Try typing \space <return> \space to proceed.%
830               }%
831             \endgroup
832           }%
833         }{%
834           \@pass@options\@currentx{#2}{#1}%
835           \global\expandafter
836           \let\csname ver@\@currname.\@currentx\endcsname\@empty
837           \InputIfFileExists
838             {\@currname.\@currentx}%

```

```

839     {}%
840     {\@missingfileerror\@currname\@currentx}%
841     \let\@unprocessedoptions\@unprocessedoptions
842     \csname\@currname.\@currentx-h@@k\endcsname
843     \expandafter\let\csname\@currname.\@currentx-h@@k\endcsname
844         \@undefined
845     \@unprocessedoptions
846 }%
847 \@ifl@ter\@currentx{#1}{#3}{%
848 }{%
849     \@latex@warning@no@line{%
850         You have requested,\on@line,
851         version\MessageBreak
852         #3' of \@cls@pkg\space #1,\MessageBreak
853         but only version\MessageBreak
854         '\csname ver@#1.\@currentx\endcsname'\MessageBreak
855         is available
856     }%
857 }%
858 \ifx\@currentx\@clsextension\let\LoadClass\@twoloadclasserror\fi
859 \@popfilename
860 \@reset@options
861 }%
862 \reserved@a
863 }

864 \def\@unknownoptionerror{%
865     \@latex@error{%
866         Unknown option '\KVO@SanitizedCurrentOption' %
867         for \@cls@pkg\space'\@currname'%
868     }{%
869         The option '\KVO@SanitizedCurrentOption' was not declared in
870         \@cls@pkg\space'\@currname', perhaps you\MessageBreak
871         misspelled its name.
872         Try typing \space <return>
873         \space to proceed.%
874     }%
875 }

876 \def\@@unprocessedoptions{%
877     \ifx\@currentx\@pkgextension
878         \@ifundefined{opt@\@currname.\@currentx}{%
879             \let\@curroptions\@empty
880         }{%
881             \expandafter\let\expandafter\@curroptions
882                 \csname opt@\@currname.\@currentx\endcsname
883         }%
884         \@for\CurrentOption:=\@curroptions\do{%
885             \ifx\CurrentOption\@empty\else\@unknownoptionerror\fi
886         }%
887     \fi
888 }

889 \def\KVO@SanitizedCurrentOption{%
890     \expandafter\strip@prefix\meaning\CurrentOption
891 }

    Normalize option list.
892 \def\KVO@normalize#1#2{%
893     \let\KVO@result\@empty
894     \KVO@splitcomma#2,\@nil
895     \let#1\KVO@result
896 }
897 \def\KVO@splitcomma#1,#2\@nil{%
898     \KVO@ifempty{#1}{-}{%

```

```

899 \KVO@checkkv#1=\@nil
900 }%
901 \KVO@ifempty{#2}{\KVO@splitcomma#2\@nil}%
902 }
903 \def\KVO@ifempty#1{%
904 \expandafter\ifx\expandafter\\\detokenize{#1}\\%
905 \expandafter\@firstoftwo
906 \else
907 \expandafter\@secondoftwo
908 \fi
909 }
910 \def\KVO@checkkv#1=#2\@nil{%
911 \KVO@ifempty{#2}{%
912 % option without value
913 \edef\KVO@x{\zap@space#1 \@empty}%
914 \ifx\KVO@x\@empty
915 % ignore empty option
916 \else
917 % append to list
918 \edef\KVO@result{%
919 \unexpanded\expandafter{\KVO@result},\KVO@x
920 }%
921 \fi
922 }{%
923 % #1: "key", #2: "value="
924 % add key part
925 \edef\KVO@result{%
926 \unexpanded\expandafter{\KVO@result},%
927 \zap@space#1 \@empty
928 }%
929 \futurelet\@let@token\KVO@checkfirsttok#2 \@nil| = \@nil|\KVO@nil
930 }%
931 }
932 \def\KVO@checkfirsttok{%
933 \ifx\@let@token\bgroup
934 % no space at start
935 \expandafter\KVO@removelastspace\expandafter=%
936 % "<value><spaceopt>= \@nil"
937 \else
938 \expandafter\KVO@checkfirstA
939 \fi
940 }
941 \def\KVO@checkfirstA#1 #2\@nil{%
942 \KVO@ifempty{#2}{%
943 \KVO@removelastspace=#1 \@nil
944 }{%
945 \KVO@ifempty{#1}{%
946 \KVO@removelastspace=#2\@nil
947 }{%
948 \KVO@removelastspace=#1 #2\@nil
949 }%
950 }%
951 }
952 \def\KVO@removelastspace#1 = \@nil|#2\KVO@nil{%
953 \KVO@ifempty{#2}{%
954 \edef\KVO@result{%
955 \unexpanded\expandafter{\KVO@result}%
956 \unexpanded\expandafter{\KVO@removegarbage#1\KVO@nil}%
957 }%
958 }{%
959 \edef\KVO@result{%
960 \unexpanded\expandafter{\KVO@result}%

```

```

961     \unexpanded{#1}%
962   }%
963 }%
964 }
965 \def\KV0@removegarbage#1= \@nil#2\KV0@nil{#1}%
    Arguments #1 and #2 are macros.
966 \def\KV0@removeelement#1#2{%
967   \begingroup
968     \toks@={}%
969     \@for\x:=#2\do{%
970       \ifx\x\@empty
971       \else
972         \ifx\x#1\relax
973         \else
974           \edef\t{\the\toks@}%
975           \ifx\t\@empty
976           \else
977             \toks@\expandafter{\the\toks@,}%
978           \fi
979           \toks@\expandafter{\the\expandafter\toks@\x}%
980         \fi
981       \fi
982     }%
983     \edef\x{\endgroup
984       \def\noexpand#2{\the\toks@}%
985     }%
986   \x
987 }

988 \let\@fileswith@pti@ns\KV0@fileswith@pti@ns
989 \ifx\@fileswith@pti@ns\@badrequireerror
990 \else
991   \let\@fileswith@pti@ns\KV0@fileswith@pti@ns
992 \fi
993 </package>

```

## 7 Installation

**CTAN.** This package is available on CTAN<sup>1</sup>:

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

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

**Unpacking.** The .dtx file is a self-extracting docstrip archive. The files are extracted by running the .dtx through plain- $\text{\TeX}$ :

```
tex kvoptions.dtx
```

**TDS.** Now the different files must be moved into the different directories in your installation TDS tree (also known as  $\text{\texttt{texmf}}$  tree):

kvoptions.sty	→	tex/latex/oberdiek/kvoptions.sty
kvoptions.pdf	→	doc/latex/oberdiek/kvoptions.pdf
example-mycolorsetup.sty	→	doc/latex/oberdiek/example-mycolorsetup.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.

<sup>1</sup><http://ftp.ctan.org/tex-archive/>

**Refresh file databases.** If your T<sub>E</sub>X distribution (teT<sub>E</sub>X, mikT<sub>E</sub>X, ...) rely on file databases, you must refresh these. For example, teT<sub>E</sub>X users run `texhash` or `mktextlsr`.

## 7.1 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 L<sup>A</sup>T<sub>E</sub>X.** The `.dtx` chooses its action depending on the format:

**plain-T<sub>E</sub>X:** Run `docstrip` and extract the files.

**L<sup>A</sup>T<sub>E</sub>X:** Generate the documentation.

If you insist on using L<sup>A</sup>T<sub>E</sub>X for `docstrip` (really, `docstrip` does not need L<sup>A</sup>T<sub>E</sub>X), 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 pdfL<sup>A</sup>T<sub>E</sub>X:

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

## 8 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 `tracefmt`, 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/](http://CTAN:macros/latex/contrib/xkeyval/)
- [9] The L<sup>A</sup>T<sub>E</sub>X3 Project, *L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> for class and package writers*, 2003/12/09.  
[CTAN:macros/latex/doc/clsguide.pdf](http://CTAN:macros/latex/doc/clsguide.pdf)

## 9 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<sup>A</sup>T<sub>E</sub>X’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<sup>A</sup>T<sub>E</sub>X. 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 meaning of `\CurrentOption` at all.

## 10 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	
\@fileswith@pti@ns	715, 988
\@unprocessedoptions	841, 876
\@SetupDriver	41, 43
\@badrequireerror	989
\@classoptionslist	474, 476, 657, 675, 757, 759
\@cls@pkg	817, 852, 867, 870
\@clsexension	156, 157, 176, 194, 195, 219, 232, 265, 293, 294, 315, 325, 346, 426, 438, 449, 472, 504, 654, 673, 731, 756, 858
\@current	116, 119, 122, 126, 129, 132, 156, 176, 177, 194, 265, 293, 333, 472, 498, 502, 504, 529, 571, 638, 644, 654, 673, 731, 797, 798, 803, 804, 807, 812, 834, 836, 838, 840, 842, 843, 847, 854, 858, 877, 878, 882
\@currname	57, 116, 119, 120, 122, 126, 129, 130, 132, 155, 177, 193, 270, 292, 333, 498, 502, 529, 571, 638, 644, 796, 798, 836, 838, 840, 842, 843, 867, 870, 878, 882
\@curroptions	639, 642, 659, 704, 879, 881, 884
\@declaredoptions	650, 678, 711
\@documentclasshook	764
\@ehc	181, 361, 410, 446
\@empty	87, 88, 343, 392, 471, 512, 557, 565, 568, 578, 582, 637, 639, 651, 666, 676, 681, 714, 733, 786, 798, 799, 836, 879, 885, 893, 913, 914, 927, 970, 975
\@expandtwoargs	490
\@fileswith@pti@ns	715, 989, 991
\@firstofone	214, 217, 285
\@firstoftwo	905
\@for	476, 507, 536, 567, 650, 675, 691, 704, 711, 749, 884, 969
\@gobble	228, 388, 589
\@gobbletwo	283
\@if@pti@ns	611
\@if@ptions	607, 804
\@ifdefinable	208
\@ifl@aded	803
\@ifl@ter	847
\@ifnextchar	246
\@ifpackageloaded	599
\@ifstar	460, 647
\@ifundefined	119, 129, 143, 170, 356, 477, 498, 508, 529, 537, 623, 638, 705, 807, 878
\@latex@error	817, 865
\@latex@warning@no@line	849
\@let@token	929, 933
\@missingfileerror	840
\@namedef	254, 304, 333
\@nameuse	571
\@one	105, 110, 112
\@nil	477, 509, 538, 560, 570, 576, 583, 586, 777, 787, 894, 897, 899, 901, 910, 929, 936, 941, 943, 946, 948, 952, 965
\@onelevel@sanitize	212, 243, 244, 489, 511
\@pass@ptions	621, 834
\@pkgextension	159, 197, 296, 350, 775, 781, 877
\@popfilename	859
\@pushfilename	795
\@removeelement	490, 721
\@reset@ptions	800, 860
\@secondoftwo	907
\@tempc	470
\@twoloadclasserror	858
\@undefined	844
\@unknownoptionerror	67, 864, 885
\@unprocessedoptions	558, 716, 841, 845
\@unusedoptionlist	491, 512, 513, 516, 518, 724, 727, 732, 733, 735
\\	256, 904
A	
\AtEndOfPackage	104, 558, 716
C	
\catcode	105, 106, 110, 111, 112, 113
\ClassError	439
\ClassInfo	233, 266, 326, 427
\ClassWarning	220, 316, 450
\csname	116, 122, 126, 132, 142, 150, 186, 187, 188, 189, 208, 238, 275, 300, 363, 371, 375, 387, 391, 414, 419, 420, 422, 423, 502, 593, 613, 624, 629, 632, 644, 666, 681, 712, 728, 744, 750, 798, 812, 836, 842, 843, 854, 882
\CurrentOption	35, 37, 38, 41, 58, 62, 63, 309, 505, 507, 509, 511, 513, 519, 524, 534, 536, 538, 541, 545, 557, 567, 568, 570, 579, 583, 650, 651, 661, 666, 675, 676, 678, 681, 704, 711, 712, 714, 722, 737, 747, 749, 750, 753, 799, 884, 885, 890
\CurrentOption@SaveLevel	741, 744, 748, 752
\CurrentOptionKey	579, 582
\CurrentOptionValue	56, 577, 578, 580, 587

<b>D</b>	
<code>\DeclareBoolOption</code> .	4, 20, 22, 138, 339
<code>\DeclareComplementaryOption</code> . . . . .	5, 169, 340
<code>\DeclareDefaultOption</code> . . . . .	5, 55, 332
<code>\DeclareOption</code> . . . . .	112, 113
<code>\DeclareStringOption</code> . . . . .	4, 26, 28, 50, 52, 245, 341
<code>\DeclareVoidOption</code> . . . . .	5, 32, 33, 34, 281
<code>\default@ds</code> . . . . .	707
<code>\define@key</code> . . . . .	115, 125, 154, 192, 262, 291, 344, 348, 413
<code>\detokenize</code> . . . . .	612, 616, 722, 724, 735, 737, 904
<code>\DisableKeyvalOption</code> . . . . .	6, 352
<code>\do</code> . . . . .	476, 507, 536, 567, 650, 675, 691, 704, 711, 749, 884, 969
<code>\documentclass</code> . . . . .	827
<code>\ds@</code> . . . . .	637
<b>E</b>	
<code>\emph</code> . . . . .	48, 86, 90, 92
<code>\endcsname</code> . . . . .	116, 122, 126, 132, 142, 150, 186, 187, 188, 189, 208, 238, 275, 300, 363, 371, 375, 387, 391, 414, 419, 420, 422, 423, 502, 593, 613, 625, 630, 632, 644, 666, 681, 712, 728, 744, 750, 798, 812, 836, 842, 843, 854, 882
<code>\endinput</code> . . . . .	590, 597, 605
<code>\ExecuteOptions</code> . . . . .	742
<b>F</b>	
<code>\futurelet</code> . . . . .	929
<b>G</b>	
<code>\gdef</code> . . . . .	515, 624, 629, 759
<b>I</b>	
<code>\ifin@</code> . . . . .	664, 679, 697
<code>\ifKV@dyn@global</code> . . . . .	369, 373, 380, 385, 389, 396, 417, 433
<code>\ifMCS@print</code> . . . . .	78
<code>\ifpdf</code> . . . . .	25
<code>\ifx</code> . . . . .	56, 87, 156, 176, 194, 213, 216, 219, 232, 256, 265, 293, 313, 315, 325, 406, 426, 438, 449, 472, 474, 504, 512, 565, 568, 578, 593, 651, 654, 673, 676, 692, 731, 733, 756, 757, 777, 779, 858, 877, 885, 904, 914, 933, 970, 972, 975, 989
<code>\in@</code> . . . . .	661
<code>\in@false</code> . . . . .	689
<code>\in@true</code> . . . . .	693, 698
<code>\InputIfFileExists</code> . . . . .	837
<b>K</b>	
<code>\KV@sp@def</code> . . . . .	587
<code>\KV@setcurrentvalue</code> . . . . .	586
<code>\KV@action@error</code> . . . . .	399
<code>\KV@action@ignore</code> . . . . .	383
<code>\KV@action@undef</code> . . . . .	367
<code>\KV@action@warning</code> . . . . .	402
<code>\KV@boolkey</code> . . . . .	155, 193, 210
<code>\KV@calldefault</code> . . . . .	553, 561
<code>\KV@checkfirstA</code> . . . . .	938, 941
<code>\KV@checkfirsttok</code> . . . . .	929, 932
<code>\KV@checkkv</code> . . . . .	899, 910
<code>\KV@CurrentOption</code> . . . . .	476, 477, 485, 489, 490
<code>\KV@DeclareStringOption</code> . . . . .	247, 249, 252
<code>\KV@disable@error</code> . . . . .	437
<code>\KV@disable@warning</code> . . . . .	448
<code>\KV@do@action</code> . . . . .	400, 403, 405
<code>\KV@ExecuteOptions</code> . . . . .	743, 746
<code>\KV@false</code> . . . . .	216, 241
<code>\KV@family</code> . . . . .	115, 154, 178, 192, 262, 291, 463
<code>\KV@fileswith@pti@ns</code> . . . . .	755, 988, 991
<code>\KV@getkey</code> . . . . .	477, 509, 538, 560
<code>\KV@ifdefinable</code> . . . . .	139, 140, 141, 184, 185, 207, 253, 284
<code>\KV@ifempty</code> . . . . .	898, 901, 903, 911, 942, 945, 953
<code>\KV@in@</code> . . . . .	678, 688
<code>\KV@nil</code> . . . . .	929, 952, 956, 965
<code>\KV@normalize</code> . . . . .	609, 622, 758, 892
<code>\KV@onefilewithoptions</code> . . . . .	763, 768, 774, 781, 794
<code>\KV@param</code> . . . . .	211, 212, 213, 216, 225, 237, 238
<code>\KV@prefix</code> . . . . .	125, 139, 140, 141, 142, 143, 150, 161, 170, 179, 184, 185, 186, 187, 188, 189, 199, 253, 254, 275, 284, 300, 304
<code>\KV@process@pti@ns</code> . . . . .	670, 686, 703
<code>\KV@process@ptions</code> . . . . .	647, 649
<code>\KV@ProcessKeyvalOptions</code> . . . . .	463, 467, 469
<code>\KV@removeelement</code> . . . . .	966
<code>\KV@removegarbage</code> . . . . .	956, 965
<code>\KV@removelastspace</code> . . . . .	935, 943, 946, 948, 952
<code>\KV@result</code> . . . . .	893, 895, 918, 919, 925, 926, 954, 955, 959, 960
<code>\KV@SanitizedCurrentOption</code> . . . . .	705, 728, 866, 869, 889
<code>\KV@setcurrents</code> . . . . .	570, 576
<code>\KV@setcurrentvalue</code> . . . . .	583, 586
<code>\KV@splitcomma</code> . . . . .	894, 897, 901
<code>\KV@temp</code> . . . . .	471, 479, 481, 499, 506, 531, 535, 552, 556, 609, 616, 622, 626, 632, 758, 760, 773, 781, 789
<code>\KV@true</code> . . . . .	213, 241
<code>\KV@use@ption</code> . . . . .	665, 680, 706, 709, 718
<code>\KV@VOID@</code> . . . . .	291, 307, 313
<code>\KV@voidkey</code> . . . . .	292, 308
<code>\KV@x</code> . . . . .	913, 914, 919
<code>\KV@xprocess@ptions</code> . . . . .	647, 672
<code>\KV@dyn@action</code> . . . . .	356, 359, 363
<code>\KV@dyn@ext</code> . . . . .	343, 346, 350, 415, 426
<code>\KV@dyn@name</code> . . . . .	342, 345, 349, 406, 415, 431



<b>L</b>		\reserved@b . . . . . 776, 783, 787	
\LoadClass . . . . .	858	<b>S</b>	
<b>M</b>		\setkeys . . . . .	44, 136, 354, 554
\makeatletter . . . . .	801	\SetupDriver . . . .	32, 33, 34, 35, 38, 40
\MCS@driver . . . . .	83	\SetupKeyvalOptions . .	3, 13, 135, 335
\MCS@emph . . . . .	87, 93	\space . . . . .	176, 817, 820, 823, 825, 829, 852, 867, 870, 872, 873
\meaning . . . . .	359, 812, 816, 890	\strip@prefix . . . .	359, 811, 816, 890
\MessageBreak . . . . .		<b>T</b>	
.. 58, 145, 146, 172, 177, 178, 179, 180, 225, 321, 359, 379, 395, 408, 432, 444, 455, 601, 602, 819, 820, 822, 823, 824, 826, 828, 851, 852, 853, 854, 870		\t . . . . .	974, 975
<b>N</b>		\textcolor . . . . .	93
\NeedsTeXFormat . . . . .	4, 98	\the 105, 106, 262, 271, 505, 524, 531, 534, 541, 545, 553, 554, 661, 748, 752, 789, 974, 977, 979, 984	
\newcommand . . . . .	40, 43, 135, 138, 169, 245, 281, 332, 352, 459	\toks . . . . .	497, 540, 541, 553, 659, 661
\newif . . . . .	142	\toks@ . . . . .	257, 259, 262, 264, 271, 499, 501, 505, 506, 523, 524, 530, 531, 534, 535, 544, 545, 554, 655, 657, 661, 788, 789, 968, 974, 977, 979, 984
\next . . . . .	283, 285, 288	\tw@ . . . . .	106, 111, 113, 497, 540, 541, 553, 659, 661
\numexpr . . . . .	748, 752	<b>U</b>	
<b>O</b>		\unexpanded . . . . .	480, 484, 919, 926, 955, 956, 960, 961
\on@line . . . . .	850	<b>X</b>	
\OptionNotUsed . . . . .	730	\x . . . . .	103, 109, 153, 164, 191, 202, 261, 278, 290, 303, 311, 313, 355, 365, 368, 384, 412, 462, 465, 563, 565, 610, 619, 660, 663, 691, 692, 696, 701, 720, 727, 808, 810, 820, 826, 969, 970, 972, 979, 983, 986
<b>P</b>		<b>Y</b>	
\PackageError . . . . .	171, 357, 407, 441	\y . . . . .	815, 816, 823, 826
\PackageInfo . . . . .		<b>Z</b>	
.. 235, 268, 328, 378, 394, 429, 591		\zap@space . . . . .	582, 786, 913, 927
\PackageWarning . . .	144, 222, 318, 452		
\PackageWarningNoLine . .	57, 594, 600		
\PassOptionsToPackage . . . . .	63, 79		
\ProcessKeyvalOptions . . . .	3, 73, 459		
\ProcessOptions . . . . .	114, 636		
\ProvidesPackage . . . . .	5, 99		
<b>R</b>			
\renewcommand . . . . .	92		
\RequirePackage . . . . .	7, 8, 83, 101		
\reserved@a . . . . .	762, 767, 786, 787, 790, 792, 802, 862		