

# The kvoptions package

Heiko Oberdiek  
<oberdiek@uni-freiburg.de>

2007/10/18 v3.0

## 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 . . . . .	3
<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 . . . . .	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
<b>3</b>	<b>Example</b>	<b>8</b>
<b>4</b>	<b>Package options</b>	<b>9</b>
4.1	Package kvoptions-patch . . . . .	9
4.2	Option debugshow . . . . .	11
<b>5</b>	<b>Limitations</b>	<b>11</b>
5.1	Compatibility . . . . .	11
5.1.1	Package kvoptions-patch vs. package xkvltxp . . . . .	11
5.2	Limitations . . . . .	11
5.2.1	Option comparisons . . . . .	11
5.2.2	Option list parsing with package kvoptions-patch . . . . .	12
<b>6</b>	<b>Implementation</b>	<b>12</b>
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<sub>E</sub>X</code>	24
6.7	Package <code>kvoptions-patch</code>	24
<b>7</b>	<b>Test</b>	<b>32</b>
7.1	Preface for standard catcode check	32
7.2	Catcode checks for loading	32
<b>8</b>	<b>Installation</b>	<b>33</b>
8.1	Download	33
8.2	Bundle installation	34
8.3	Package installation	34
8.4	Refresh file name databases	34
8.5	Some details for the interested	34
<b>9</b>	<b>References</b>	<b>35</b>
<b>10</b>	<b>History</b>	<b>35</b>
	[0000/00/00 v0.0]	35
	[2004/02/22 v1.0]	36
	[2006/02/16 v2.0]	36
	[2006/02/20 v2.1]	36
	[2006/06/01 v2.2]	36
	[2006/08/17 v2.3]	36
	[2006/08/22 v2.4]	36
	[2007/04/11 v2.5]	36
	[2007/05/06 v2.6]	36
	[2007/06/11 v2.7]	36
	[2007/10/02 v2.8]	36
	[2007/10/11 v2.9]	37
	[2007/10/18 v3.0]	37
<b>11</b>	<b>Index</b>	<b>37</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 L<sup>A</sup>T<sub>E</sub>X's package and class `options`:

Package <code>keyval</code>	Package <code>kvoptions</code>	L <sup>A</sup> T <sub>E</sub> 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<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.

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<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>	(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

<code>\DeclareDefaultOption{⟨code⟩}</code>
--

This command does not define a specific 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. 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<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` 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

<code>\DisableKeyvalOption [⟨options⟩] {⟨family⟩} {⟨key⟩}</code>		
<code>⟨options⟩:</code>		
<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 `\<prefix>` that can be configured by `\SetupKeyvalOptions`.

Declare <i>&lt;key&gt;</i>	Defined macro	Description
<code>\DeclareStringOption</code>	<code>\&lt;prefix&gt;\&lt;key&gt;</code>	holds the string
<code>\DeclareBoolOption</code>	<code>\if\&lt;prefix&gt;\&lt;key&gt;</code> <code>\&lt;prefix&gt;\&lt;key&gt;false</code> <code>\&lt;prefix&gt;\&lt;key&gt;true</code>	boolean switch disable switch enable switch
<code>\DeclareComplementaryOption</code>	<code>\&lt;prefix&gt;\&lt;key&gt;false</code> <code>\&lt;prefix&gt;\&lt;key&gt;true</code>	enable parent switch disable parent switch
<code>\DeclareVoidOption</code>	<code>\&lt;prefix&gt;\&lt;key&gt;</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}[2007/10/18 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  $\text{\LaTeX}$  is unhappy if a package is loaded later again with options that are previously not given. Thus package and class authors, stay with `\RequirePackage{kvoptions}` without options.

Option `patch` loads package `kvoptions-patch`.

### 4.1 Package `kvoptions-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 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 Package `kvoptions-patch` vs. package `xkvltxp`

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

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

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

The other way should work, too.

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

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

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

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

### 5.2 Limitations

#### 5.2.1 Option comparisons

In some situations L<sup>A</sup>T<sub>E</sub>X compares option lists, e.g. option clash check, `\@ifpackagewith`, or `\@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 package `kvoptions-patch`

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

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

Example:

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

Result:

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

## 6 Implementation

### 6.1 Preamble

```
98 <*package>
```

**Reload check and identification.** Reload check, especially if the package is not used with  $\text{\LaTeX}$ .

```
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 [2007/10/18 v3.0 Keyval support for 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{45}{12}% -
172 \TMP@EnsureCode{46}{12}% .
173 \TMP@EnsureCode{47}{12}% /
174 \TMP@EnsureCode{58}{12}% :
175 \TMP@EnsureCode{61}{12}% =
176 \TMP@EnsureCode{62}{12}% >
177 \TMP@EnsureCode{94}{7}% ^ (superscript)
178 \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`.

```

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

```

```
181 }{}
```

### Provide macros for plain-TeX.

```
182 \@ifundefined{@onelevel@sanitize}{%
183   \def\@onelevel@sanitize#1{%
184     \edef#1{\expandafter\strip@prefix\meaning#1}%
185   }%
186 }{}
187 \@ifundefined{strip@prefix}{%
188   \def\strip@prefix#1>{%
189 }{}
190 \@ifundefined{@x@protect}{%
191   \def\@x@protect#1\fi#2#3{%
192     \fi\protect#1%
193   }%
194   \let\@typeset@protect\relax
195 }{}
196 \@ifundefined{@currname}{%
197   \def\@currname{}%
198 }{}
199 \@ifundefined{@currentx}{%
200   \def\@currentx{}%
201 }{}
```

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

```
202 \DeclareOption{debugshow}{\catcode\@ne=9 }
203 \DeclareOption{patch}{%
204   \AtEndOfPackage{%
205     \RequirePackage{kvoptions-patch}[2007/10/18]%
206   }%
207 }
```

Optionen auswerten:

```
208 \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.

```
209 \define@key{KV@}{family}{%
210   \expandafter\edef\csname KV@family@%
211     \@currname.\@currentx\endcsname{#1}%
212 }
213 \def\KV@family{%
214   \@ifundefined{KV@family@\@currname.\@currentx}{%
215     \@currname
216   }{%
217     \csname KV@family@\@currname.\@currentx\endcsname
218   }%
219 }
```

`\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.

```

220 \define@key{KV0}{prefix}{%
221   \expandafter\edef\csname KV0@prefix@%
222     \@currname.\@current\endcsname{#1}%
223 }
224 \def\KV0@prefix{%
225   \@ifundefined{KV0@prefix@\@currname.\@current}{%
226     \@currname %
227   }{%
228     \csname KV0@prefix@\@currname.\@current\endcsname
229   }%
230 }
```

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

```

231 \newcommand*{\SetupKeyvalOptions}{%
232   \setkeys{KV0}%
233 }
```

## 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.

```

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

```

262     }%
263 }%
264 }%
265 }

```

`\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.

```

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

```

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

```

289     \begingroup
290     \edef\x{\endgroup
291       \noexpand\define@key{\KV0@family}{#1}[true]{%
292         \noexpand\KV0@boolkey{\@currname}%
293         \ifx\@currentx\@clsextension
294           \noexpand\@clsextension
295         \else
296           \noexpand\@pkgextension
297         \fi
298         {\KV0@prefix}{#1}{####1}%
299       }%
300     }%
301     \x
302   }%
303 }%
304 }%
305 }

```

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

```

306 \def\KV0@ifdefinable#1{%
307   \expandafter\@ifdefinable\csname #1\endcsname
308 }

```

`\KV0@boolkey` We check explicitly for `true` and `false` to prevent the user from accidentally calling other macros.



```

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

```

```

309 \def\KV0@boolkey#1#2#3#4#5{%
310   \edef\KV0@param{#5}%
311   \@onelevel@sanitize\KV0@param
312   \ifx\KV0@param\KV0@true
313     \expandafter\@firstofone
314   \else
315     \ifx\KV0@param\KV0@false
316       \expandafter\expandafter\expandafter\@firstofone
317     \else
318       \ifx#2\@clsextension
319         \expandafter\ClassWarning
320       \else
321         \expandafter\PackageWarning
322       \fi
323     {#1}{%
324       Value '\KV0@param' is not supported by\MessageBreak
325       option '#4'%
326     }%
327     \expandafter\expandafter\expandafter\@gobble
328   \fi
329 \fi
330 {%
331   ^^A\ifx#2\@clsextension
332   ^^A \expandafter\ClassInfo
333   ^^A\else
334   ^^A \expandafter\PackageInfo
335   ^^A\fi
336   ^^A{#1}{[option] #4=\KV0@param}%
337   \csname#3#4\KV0@param\endcsname
338 }%
339 }

```

\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).

```

340 \def\KV0@true{true}
341 \def\KV0@false{false}
342 \@onelevel@sanitize\KV0@true
343 \@onelevel@sanitize\KV0@false

```

### 6.2.3 \DeclareStringOption

\DeclareStringOption

```

344 \newcommand*\DeclareStringOption[2][ ]{%
345   \@ifnextchar[%
346     \KV0@DeclareStringOption{#1}{#2}%
347   ]{%
348     \KV0@DeclareStringOption{#1}{#2}{ }[%
349   ]%
350 }

```

\KV0@DeclareStringOption

```

351 \def\KV0@DeclareStringOption#1#2#3[#4]{%
352   \KV0@ifdefinable{\KV0@prefix#2}{%
353     \@namedef{\KV0@prefix#2}{#1}%
354     \begingroup
355     \ifx\#3\%

```

```

356     \toks@{%
357     \else
358     \toks@{[#{4}]}%
359     \fi
360     \edef\x{\endgroup
361     \noexpand\define@key{\KVO@family}{#2}\the\toks@{%
362     ^^A\begingroup
363     ^^A \toks@{####1}%
364     ^^A \ifx\@current\@clsextension
365     ^^A \noexpand\ClassInfo
366     ^^A \else
367     ^^A \noexpand\PackageInfo
368     ^^A \fi
369     ^^A {\@currname}{%
370     ^^A [option] #2={\noexpand\the\toks@}%
371     ^^A }%
372     ^^A\endgroup
373     \noexpand\def
374     \expandafter\noexpand\csname\KVO@prefix#2\endcsname{####1}%
375     }%
376     }%
377     \x
378     }%
379 }

```

#### 6.2.4 \DeclareVoidOption

\DeclareVoidOption

```

380 \newcommand*{\DeclareVoidOption}[1]{%
381   \begingroup
382   \let\next\@gobbletwo
383   \KVO@ifdefinable{\KVO@prefix#1}{%
384     \let\next\@firstofone
385   }%
386   \expandafter\endgroup
387   \next{%
388     \begingroup
389     \edef\x{\endgroup
390       \noexpand\define@key{\KVO@family}{#1}[\KVO@VOID@]{%
391         \noexpand\KVO@voidkey{\@currname}%
392         \ifx\@current\@clsextension
393           \noexpand\@clsextension
394         \else
395           \noexpand\@pkgextension
396         \fi
397         {#1}%
398         {####1}%
399         \expandafter\noexpand\csname\KVO@prefix#1\endcsname
400       }%
401     }%
402     \x
403     \@namedef{\KVO@prefix#1}%
404   }%
405 }
406 \def\KVO@VOID@{@VOID@}

```

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

```

408 \def\CurrentOption{#3}%
409 \begingroup
410   \def\x{#4}%
411 \expandafter\endgroup
412 \ifx\x\KV0@VOID@
413 \else
414   \ifx#2\@clsextension
415     \expandafter\ClassWarning
416   \else
417     \expandafter\PackageWarning
418   \fi
419   {#1}{%
420     Unexpected value for option ‘#3’\MessageBreak
421     is ignored%
422   }%
423 \fi
424 ^^A\ifx#2\@clsextension
425 ^^A \expandafter\ClassInfo
426 ^^A\else
427 ^^A \expandafter\PackageInfo
428 ^^A\fi
429 ^^A{#1}{[option] #3}%
430 }

```

### 6.2.5 \DeclareDefaultOption

\DeclareDefaultOption

```

431 \newcommand*{\DeclareDefaultOption}{%
432   \@namedef{KV0@default@\@currname.\@current}{%
433 }

```

## 6.3 Dynamic options

### 6.3.1 \DisableKeyvalOption

```

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

```

```

461     }{%
462     \csname KV0@action@\KV0dyn@action\endcsname{#2}{#3}%
463     }%
464     \x
465 }
466 \def\KV0@action@undef#1#2{%
467     \edef\x{\endgroup
468         \ifKV0dyn@global\global\fi
469         \let
470         \expandafter\noexpand\csname KV@#1@#2\endcsname
471         \relax
472         \ifKV0dyn@global\global\fi
473         \let
474         \expandafter\noexpand\csname KV@#1@#2@default\endcsname
475         \relax
476     }%
477     ^^A\PackageInfo{kvoptions}{%
478     ^^A [option] key ‘#2’ of family ‘#1’\MessageBreak
479     ^^A is disabled (undef, \ifKV0dyn@global\global\else local\fi)%
480     ^^A}%
481 }
482 \def\KV0@action@ignore#1#2{%
483     \edef\x{\endgroup
484         \ifKV0dyn@global\global\fi
485         \let
486         \expandafter\noexpand\csname KV@#1@#2\endcsname
487         \@gobble
488         \ifKV0dyn@global\global\fi
489         \let
490         \expandafter\noexpand\csname KV@#1@#2@default\endcsname
491         \@empty
492     }%
493     ^^A\PackageInfo{kvoptions}{%
494     ^^A [option] key ‘#2’ of family ‘#1’\MessageBreak
495     ^^A is disabled (ignore, \ifKV0dyn@global\global\else local\fi)%
496     ^^A}%
497 }
498 \def\KV0@action@error{%
499     \KV0@do@action{error}%
500 }
501 \def\KV0@action@warning{%
502     \KV0@do@action{warning}%
503 }

#1 error or warning
#2 <family>
#3 <key>
504 \def\KV0@do@action#1#2#3{%
505     \ifx\KV0dyn@name\relax
506         \PackageError{kvoptions}{%
507             Action type ‘#1’ needs package/class name\MessageBreak
508             for key ‘#3’ in family ‘#2’%
509         }\@ehc
510     \else
511         \edef\x{\endgroup
512             \noexpand\define@key{#2}{#3}[] {%
513                 \expandafter\noexpand\csname KV0@disable@#1\endcsname
514                 {\KV0dyn@name}\noexpand\KV0dyn@ext{#3}%
515             }%
516             \ifKV0dyn@global
517                 \global\let
518                 \expandafter\noexpand\csname KV@#2@#3\endcsname
519                 \expandafter\noexpand\csname KV@#2@#3\endcsname

```

```

520         \global\let
521         \expandafter\noexpand\csname KV@#2@#3@default\endcsname
522         \expandafter\noexpand\csname KV@#2@#3@default\endcsname
523     \fi
524 }%
525 ^^A\ifx\KV0dyn@ext\@clsextension
526 ^^A \expandafter\ClassInfo
527 ^^A\else
528 ^^A \expandafter\PackageInfo
529 ^^A\fi
530 ^^A{\KV0dyn@name}{%
531 ^^A [option] key ‘#3’ of family ‘#2’\MessageBreak
532 ^^A is disabled (#1, \ifKV0dyn@global global\else local\fi)%
533 ^^A}%
534 \fi
535 }
536 \def\KV0@disable@error#1#2#3{%
537 \ifx#2\@clsextension
538 \expandafter\ClassError
539 \else
540 \expandafter\PackageError
541 \fi
542 {#1}{%
543 Option ‘#3’ is given too late,\MessageBreak
544 now the option is ignored%
545 }\@ehc
546 }
547 \def\KV0@disable@warning#1#2#3{%
548 \ifx#2\@clsextension
549 \expandafter\ClassWarning
550 \else
551 \expandafter\PackageWarning
552 \fi
553 {#1}{%
554 Option ‘#3’ is already consumed\MessageBreak
555 and has no effect%
556 }%
557 }

```

## 6.4 Process options

### 6.5 \ProcessKeyvalOptions

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

```

558 \newcommand*{\ProcessKeyvalOptions}{%
559 \@ifstar{%
560 \begingroup
561 \edef\x{\endgroup
562 \noexpand\KV0@ProcessKeyvalOptions{\KV0@family}%
563 }%
564 \x
565 }%
566 \KV0@ProcessKeyvalOptions
567 }

568 \def\KV0@ProcessKeyvalOptions#1{%
569 \let\@tempc\relax
570 \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).

```

571 \ifx\@currentx\@clsextension
572 \else
573 \ifx\@classoptionslist\relax

```

```

574 \else
575 \for\KV0@CurrentOption:=\@classoptionslist\do{%
576 \ifundefined{KV@#1@\expandafter\KV0@getkey
577 \KV0@CurrentOption=\@nil}{%
578 }{%
579 \ifx\KV0@Patch Y%
580 \edef\KV0@temp{%
581 \etex@unexpanded\expandafter{%
582 \KV0@temp
583 }%
584 ,%
585 \etex@unexpanded\expandafter{%
586 \KV0@CurrentOption
587 }%
588 ,%
589 }%
590 \@onelevel@sanitize\KV0@CurrentOption
591 \else
592 \edef\KV0@temp{%
593 \KV0@temp
594 ,%
595 \KV0@CurrentOption
596 ,%
597 }%
598 \fi
599 \@expandtwoargs\@removeelement\KV0@CurrentOption
600 \@unusedoptionlist\@unusedoptionlist
601 }%
602 }%
603 \fi
604 \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`.

```

605 \begingroup
606 \toks\tw@{}%
607 \ifundefined{opt@\@currname.\@current}{%
608 \toks@\expandafter{\KV0@temp}%
609 }{%
610 \toks@\expandafter\expandafter\expandafter{%
611 \csname opt@\@currname.\@current\endcsname
612 }%
613 \ifx\@current\@clsextension
614 \edef\CurrentOption{\the\toks@}%
615 \toks@\expandafter{\KV0@temp}%
616 \@for\CurrentOption:=\CurrentOption\do{%
617 \ifundefined{%
618 KV@#1@\expandafter\KV0@getkey\CurrentOption=\@nil
619 }{%

```

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

```

620 \ifx\KV0@Patch Y%
621 \@onelevel@sanitize\CurrentOption
622 \fi
623 \ifx\@unusedoptionlist\@empty
624 \global\let\@unusedoptionlist\CurrentOption
625 \else
626 \expandafter\expandafter\expandafter\gdef
627 \expandafter\expandafter\expandafter\@unusedoptionlist
628 \expandafter\expandafter\expandafter{%
629 \expandafter\@unusedoptionlist
630 \expandafter,\CurrentOption

```

```

631         }%
632     \fi
633 }{%
634     \toks@\expandafter{%
635         \the\expandafter\toks@\expandafter,\CurrentOption
636     }%
637 }%
638 }%
639 \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.

```

640     \@ifundefined{KV0@default@{\currname.\@current}}{%
641         \toks@\expandafter\expandafter\expandafter{%
642             \expandafter\KV0@temp\the\toks@
643         }%
644     }{%
645         \edef\CurrentOption{\the\toks@}%
646         \toks@\expandafter{\KV0@temp}%
647         \@for\CurrentOption:=\CurrentOption\do{%
648             \@ifundefined{%
649                 KV@#1@\expandafter\KV0@getkey\CurrentOption=\@nil
650             }{%
651                 \toks\tw@\expandafter{%
652                     \the\toks@\expandafter\tw@\expandafter,\CurrentOption
653                 }%
654             }{%
655                 \toks@\expandafter{%
656                     \the\expandafter\toks@\expandafter,\CurrentOption
657                 }%
658             }%
659         }%
660     }%
661 \fi
662 }%
663 \edef\KV0@temp{\endgroup
664     \noexpand\KV0@calldefault{\the\toks\tw@}%
665     \noexpand\setkeys{#1}{\the\toks@}%
666 }%
667 \KV0@temp

```

Some cleanup of `\ProcessOptions`.

```

668 \let\CurrentOption\@empty
669 \AtEndOfPackage{\let\@unprocessedoptions\relax}%
670 }

```

### 6.5.1 Helper macros

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

```

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

```

`\KV0@calldefault`

```

672 \def\KV0@calldefault#1{%
673     \begingroup
674     \def\x{#1}%
675     \expandafter\endgroup
676     \ifx\x\@empty
677     \else
678         \@for\CurrentOption:=#1\do{%
679             \ifx\CurrentOption\@empty
680             \else
681                 \expandafter\KV0@setcurrents\CurrentOption=\@nil

```

```

682      \@nameuse{KV0@default@\@currname.\@current}%
683    \fi
684  }%
685 \fi
686 }

```

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

```

687 \def\KV0@setcurrents#1=#2\@nil{
688   \def\CurrentOptionValue{#2}%
689   \ifx\CurrentOptionValue\@empty
690     \let\CurrentOptionKey\CurrentOption
691     \let\CurrentOptionValue\relax
692   \else
693     \edef\CurrentOptionKey{\zap@space#1 \@empty}%
694     \expandafter\KV0@setcurrentvalue\CurrentOption\@nil
695   \fi
696 }

```

`\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.

```

697 \def\KV0@setcurrentvalue#1=#2\@nil{%
698   \KV@@sp@def\CurrentOptionValue{#2}%
699 }

```

## 6.6 plain-TeX

Disable L<sup>A</sup>T<sub>E</sub>X stuff.

```

700 \begingroup\expandafter\expandafter\expandafter\endgroup
701 \expandafter\ifx\csname documentclass\endcsname\relax
702   \def\ProcessKeyvalOptions{%
703     \@ifstar{}\@gobble
704   }%
705 \fi
706 \KV0@AtEnd
707 \</package>

```

## 6.7 Package `kvoptions-patch`

```

708 \<*patch>
709 \NeedsTeXFormat{LaTeX2e}
710 \expandafter\edef\csname KV0@AtEnd\endcsname{%
711   \catcode64 \the\catcode64\relax
712 }
713 \catcode64 11 % @
714 \def\TMP@EnsureCode#1#2{%
715   \edef\KV0@AtEnd{%
716     \KV0@AtEnd
717     \catcode#1 \the\catcode#1\relax
718   }%
719   \catcode#1 #2\relax
720 }
721 \TMP@EnsureCode{39}{12}% '
722 \TMP@EnsureCode{40}{12}% (
723 \TMP@EnsureCode{41}{12}% )
724 \TMP@EnsureCode{43}{12}% +
725 \TMP@EnsureCode{44}{12}% ,
726 \TMP@EnsureCode{45}{12}% -
727 \TMP@EnsureCode{46}{12}% .
728 \TMP@EnsureCode{47}{12}% /
729 \TMP@EnsureCode{58}{12}% :
730 \TMP@EnsureCode{60}{12}% <

```



```

731 \TMP@EnsureCode{61}{12}% =
732 \TMP@EnsureCode{62}{12}% >
733 \TMP@EnsureCode{91}{12}% [
734 \TMP@EnsureCode{93}{12}% ]
735 \TMP@EnsureCode{96}{12}% '
736 \TMP@EnsureCode{124}{12}% |
737 \edef\KVO@AtEnd{%
738   \KVO@AtEnd
739   \noexpand\endinput
740 }
741 \ProvidesPackage{kvoptions-patch}%
742 [2007/10/18 v3.0 LaTeX patch for keyval options (H0)]%

  Check for  $\varepsilon$ -TeX.
743 \begingroup\expandafter\expandafter\expandafter\endgroup
744 \expandafter\ifx\csname eTeXversion\endcsname\relax
745   \PackageWarningNoLine{kvoptions-patch}{%
746     Package loading is aborted, because e-TeX is missing%
747   }%
748   \expandafter\KVO@AtEnd
749 \fi

  Package etexcmds for \etex@unexpanded.
750 \RequirePackage{etexcmds}[2007/09/09]
751 \ifetex@unexpanded
752 \else
753   \PackageError{kvoptions-patch}{%
754     Could not find eTeX's \string\unexpanded.\MessageBreak
755     Try adding \string\RequirePackage\string{etexcmds\string} %
756     before \string\documentclass%
757   }\@ehd
758   \expandafter\KVO@AtEnd
759 \fi

  Check for package xkvltxp.
760 \@ifpackageloaded{xkvltxp}{%
761   \PackageWarningNoLine{kvoptions}{%
762     Option 'patch' cannot be used together with\MessageBreak
763     package 'xkvltxp' that is already loaded.\MessageBreak
764     Therefore package loading is aborted%
765   }%
766   \KVO@AtEnd
767 }{}

768 \def\@if@ptions#1#2#3{%
769   \begingroup
770   \KVO@normalize\KVO@temp{#3}%
771   \edef\x{\endgroup
772     \noexpand\@if@ptions{%
773       \detokenize\expandafter\expandafter\expandafter{%
774         \csname opt@#2.#1\endcsname
775       }%
776     }{%
777       \detokenize\expandafter{\KVO@temp}%
778     }%
779   }%
780   \x
781 }

782 \def\@pass@ptions#1#2#3{%
783   \KVO@normalize\KVO@temp{#2}%
784   \@ifundefined{opt@#3.#1}{%
785     \expandafter\gdef\csname opt@#3.#1%
786       \expandafter\endcsname\expandafter{%
787       \KVO@temp
788     }%

```

```

789 }{%
790   \expandafter\gdef\csname opt@#3.#1%
791     \expandafter\expandafter\expandafter\endcsname
792     \expandafter\expandafter\expandafter{%
793       \csname opt@#3.#1\expandafter\endcsname\expandafter,\KVO@temp
794     }%
795 }%
796 }

797 \def\ProcessOptions{%
798   \let\ds@\@empty
799   \@ifundefined{opt@\@currname.\@currentx}{%
800     \let\@curroptions\@empty
801   }{%
802     \expandafter\expandafter\expandafter\def
803     \expandafter\expandafter\expandafter\@curroptions
804     \expandafter\expandafter\expandafter{%
805       \csname opt@\@currname.\@currentx\endcsname
806     }%
807   }%
808   \@ifstar\KVO@xprocessoptions\KVO@processoptions
809 }

810 \def\KVO@processoptions{%
811   \@for\CurrentOption:=\@declaredoptions\do{%
812     \ifx\CurrentOption\@empty
813     \else
814       \begingroup
815         \ifx\@currentx\@clsextension
816           \toks@{}%
817         \else
818           \toks@\expandafter{\@classoptionslist,%}
819         \fi
820         \toks\tw@\expandafter{\@curroptions}%
821         \edef\x{\endgroup
822           \noexpand\in@{,\CurrentOption,}{,\the\toks@\the\toks\tw@,%}
823         }%
824         \x
825         \ifin@
826           \KVO@use@option
827           \expandafter\let\csname ds@\CurrentOption\endcsname\@empty
828         \fi
829       \fi
830     }%
831   \KVO@processoptions
832 }

833 \def\KVO@xprocessoptions{%
834   \ifx\@currentx\@clsextension
835   \else
836     \@for\CurrentOption:=\@classoptionslist\do{%
837       \ifx\CurrentOption\@empty
838       \else
839         \KVO@in@\CurrentOption\@declaredoptions
840         \ifin@
841           \KVO@use@option
842           \expandafter\let\csname ds@\CurrentOption\endcsname\@empty
843         \fi
844       \fi
845     }%
846   \fi
847   \KVO@processoptions
848 }

849 \def\KVO@in@#1#2{%

```

```

850 \in@false
851 \begingroup
852 \@for\x:=#2\do{%
853 \ifx\x#1\relax
854 \in@true
855 \fi
856 }%
857 \edef\x{\endgroup
858 \ifin@
859 \noexpand\in@true
860 \fi
861 }%
862 \x
863 }

864 \def\KV@process@pti@ns{%
865 \@for\CurrentOption:=\@curroptions\do{%
866 \@ifundefined{ds@\KV@SanitizedCurrentOption}{%
867 \KV@use@option
868 \default@ds
869 }%
870 \KV@use@option
871 }%
872 \@for\CurrentOption:=\@declaredoptions\do{%
873 \expandafter\let\csname ds@\CurrentOption\endcsname\relax
874 }%
875 \let\CurrentOption\@empty
876 \let\@fileswith@pti@ns\@fileswith@pti@ns
877 \AtEndOfPackage{\let\@unprocessedoptions\relax}%
878 }

879 \def\KV@use@option{%
880 \begingroup
881 \edef\x{\endgroup
882 \noexpand\@removeelement{%
883 \detokenize\expandafter{\CurrentOption}%
884 }{%
885 \detokenize\expandafter{\@unusedoptionlist}%
886 }%
887 }%
888 \x\@unusedoptionlist
889 \csname ds@\KV@SanitizedCurrentOption\endcsname
890 }

891 \def\OptionNotUsed{%
892 \ifx\@currentx\@clsextension
893 \xdef\@unusedoptionlist{%
894 \ifx\@unusedoptionlist\@empty
895 \else
896 \detokenize\expandafter{\@unusedoptionlist},}%
897 \fi
898 \detokenize\expandafter{\CurrentOption}%
899 }%
900 \fi
901 }

Variant of \ExecuteOptions that better protects \CurrentOption.
902 \def\CurrentOption@SaveLevel{0}
903 \def\ExecuteOptions{%
904 \expandafter\KV@ExecuteOptions
905 \csname CurrentOption@CurrentOption@SaveLevel\endcsname
906 }
907 \def\KV@ExecuteOptions#1#2{%
908 \let#1\CurrentOption
909 \edef\CurrentOption@SaveLevel{%

```

```

910 \the\numexpr\CurrentOption@SaveLevel+1%
911 }%
912 \@for\CurrentOption:=#2\do{%
913 \csname ds@\CurrentOption\endcsname
914 }%
915 \edef\CurrentOption@SaveLevel{%
916 \the\numexpr\CurrentOption@SaveLevel-1%
917 }%
918 \let\CurrentOption#1%
919 }

920 \def\KVO@fileswith@ptions#1[#2]#3[#4]{%
921 \ifx#1\@clsextension
922 \ifx\@classoptionslist\relax
923 \KVO@normalize\KVO@temp{#2}%
924 \expandafter\gdef\expandafter\@classoptionslist\expandafter{%
925 \KVO@temp
926 }%
927 \def\reserved@a{%
928 \KVO@onefilewithoptions{#3}[#{#2}][#{#4}]#1%
929 \@documentclasshook
930 }%
931 \else
932 \def\reserved@a{%
933 \KVO@onefilewithoptions{#3}[#{#2}][#{#4}]#1%
934 }%
935 \fi
936 \else
937 \begingroup
938 \let\KVO@temp\relax
939 \let\KVO@onefilewithoptions\relax
940 \let\@pkgextension\relax
941 \def\reserved@b##1,{%
942 \ifx\@nil##1\relax
943 \else
944 \ifx\relax##1\relax
945 \else
946 \KVO@onefilewithoptions{##1}[{\KVO@temp}][#{#4}]%
947 \@pkgextension
948 \fi
949 \expandafter\reserved@b
950 \fi
951 }%
952 \edef\reserved@a{\zap@space#3 \@empty}%
953 \edef\reserved@a{\expandafter\reserved@b\reserved@a,\@nil,}%
954 \toks@{#2}%
955 \def\KVO@temp{\the\toks@}%
956 \edef\reserved@a{\endgroup \reserved@a}%
957 \fi
958 \reserved@a
959 }

960 \def\KVO@onefilewithoptions#1[#2][#3]#4{%
961 \@pushfilename
962 \xdef\@currname{#1}%
963 \global\let\@current#4%
964 \expandafter\let\csname\@currname.\@current-h@k\endcsname\@empty
965 \let\CurrentOption\@empty
966 \@reset@ptions
967 \makeatletter
968 \def\reserved@a{%
969 \@ifl@aded\@current{#1}{%
970 \@if@ptions\@current{#1}{#2}{%
971 }{%

```

```

972 \begingroup
973 \@ifundefined{opt@#1.\@currentx}{%
974 \def\x{%
975 }{%
976 \edef\x{%
977 \expandafter\expandafter\expandafter\strip@prefix
978 \expandafter\meaning\csname opt@#1.\@currentx\endcsname
979 }%
980 }%
981 \def\y{#2}%
982 \edef\y{\expandafter\strip@prefix\meaning\y}%
983 \@latex@error{Option clash for \@cls@pkg\space #1}{%
984 The package #1 has already been loaded %
985 with options:\MessageBreak
986 \space\space[\x]\MessageBreak
987 There has now been an attempt to load it %
988 with options:\MessageBreak
989 \space\space[\y]\MessageBreak
990 Adding the global options:\MessageBreak
991 \space\space
992 \x,\y\MessageBreak
993 to your \noexpand\documentclass declaration may fix this.%
994 \MessageBreak
995 Try typing \space <return> \space to proceed.%
996 }%
997 \endgroup
998 }%
999 }{%
1000 \@pass@options\@currentx{#2}{#1}%
1001 \global\expandafter
1002 \let\csname ver@\@currname.\@currentx\endcsname\@empty
1003 \InputIfFileExists
1004 {\@currname.\@currentx}%
1005 }{%
1006 {\@missingfileerror\@currname\@currentx}%
1007 \let\@unprocessedoptions\@unprocessedoptions
1008 \csname\@currname.\@currentx-h@@k\endcsname
1009 \expandafter\let\csname\@currname.\@currentx-h@@k\endcsname
1010 \undefined
1011 \@unprocessedoptions
1012 }%
1013 \@ifl@ter\@currentx{#1}{#3}{%
1014 }{%
1015 \@latex@warning@no@line{%
1016 You have requested,\on@line, %
1017 version\MessageBreak
1018 #3' of \@cls@pkg\space #1,\MessageBreak
1019 but only version\MessageBreak
1020 '\csname ver@#1.\@currentx\endcsname'\MessageBreak
1021 is available%
1022 }%
1023 }%
1024 \ifx\@currentx\@clsextension\let\LoadClass\@twoloadclasserror\fi
1025 \popfilename
1026 \@reset@options
1027 }%
1028 \reserved@a
1029 }

1030 \def\@unknownoptionerror{%
1031 \@latex@error{%
1032 Unknown option '\KVO@SanitizedCurrentOption' %
1033 for \@cls@pkg\space'\@currname'%

```

```

1034 }{%
1035     The option '\KV0@SanitizedCurrentOption' was not declared in %
1036     \@cls@pkg\space'\@currname', perhaps you\MessageBreak
1037     misspelled its name. %
1038     Try typing \space <return> %
1039     \space to proceed.%
1040 }%
1041 }

1042 \def\@@unprocessedoptions{%
1043     \ifx\@current\@pkgextension
1044         \ifundefined{opt@\@currname.\@current}{%
1045             \let\@curroptions\@empty
1046         }{%
1047             \expandafter\let\expandafter\@curroptions
1048                 \csname opt@\@currname.\@current\endcsname
1049         }%
1050         \@for\CurrentOption:=\@curroptions\do{%
1051             \ifx\CurrentOption\@empty\else\@unknownoptionerror\fi
1052         }%
1053     \fi
1054 }

1055 \def\KV0@SanitizedCurrentOption{%
1056     \expandafter\strip@prefix\meaning\CurrentOption
1057 }

    Normalize option list.
1058 \def\KV0@normalize#1#2{%
1059     \let\KV0@result\@empty
1060     \KV0@splitcomma#2,\@nil
1061     \let#1\KV0@result
1062 }
1063 \def\KV0@splitcomma#1,#2\@nil{%
1064     \KV0@ifempty{#1}{}%
1065     \KV0@checkkv#1=\@nil
1066 }%
1067 \KV0@ifempty{#2}{%\KV0@splitcomma#2\@nil}%
1068 }
1069 \def\KV0@ifempty#1{%
1070     \expandafter\ifx\expandafter\\detokenize{#1}\\%
1071         \expandafter\@firstoftwo
1072     \else
1073         \expandafter\@secondoftwo
1074     \fi
1075 }
1076 \def\KV0@checkkv#1=#2\@nil{%
1077     \KV0@ifempty{#2}{%
1078         % option without value
1079         \edef\KV0@x{\zap@space#1 \@empty}%
1080         \ifx\KV0@x\@empty
1081             % ignore empty option
1082         \else
1083             % append to list
1084             \edef\KV0@result{%
1085                 \etex@unexpanded\expandafter{\KV0@result},\KV0@x
1086             }%
1087         \fi
1088     }{%
1089         % #1: "key", #2: "value="
1090         % add key part
1091         \edef\KV0@result{%
1092             \etex@unexpanded\expandafter{\KV0@result},%
1093             \zap@space#1 \@empty

```

```

1094 }%
1095 \futurelet\@let@token\KV0@checkfirsttok#2 \@nil| = \@nil|\KV0@nil
1096 }%
1097 }
1098 \def\KV0@checkfirsttok{%
1099 \ifx\@let@token\bgroup
1100 % no space at start
1101 \expandafter\KV0@removelastspace\expandafter=%
1102 % "<value><spaceopt>= \@nil"
1103 \else
1104 \expandafter\KV0@checkfirstA
1105 \fi
1106 }
1107 \def\KV0@checkfirstA#1 #2\@nil{%
1108 \KV0@ifempty{#2}{%
1109 \KV0@removelastspace=#1 \@nil
1110 }{%
1111 \KV0@ifempty{#1}{%
1112 \KV0@removelastspace=#2\@nil
1113 }{%
1114 \KV0@removelastspace=#1 #2\@nil
1115 }%
1116 }%
1117 }
1118 \def\KV0@removelastspace#1 = \@nil|#2\KV0@nil{%
1119 \KV0@ifempty{#2}{%
1120 \edef\KV0@result{%
1121 \etex@unexpanded\expandafter{\KV0@result}%
1122 \etex@unexpanded\expandafter{\KV0@removegarbage#1\KV0@nil}%
1123 }%
1124 }{%
1125 \edef\KV0@result{%
1126 \etex@unexpanded\expandafter{\KV0@result}%
1127 \etex@unexpanded{#1}%
1128 }%
1129 }%
1130 }
1131 \def\KV0@removegarbage#1= \@nil|#2\KV0@nil{#1}%

```

Arguments #1 and #2 are macros.

```

1132 \def\KV0@removeelement#1#2{%
1133 \begingroup
1134 \toks@={}%
1135 \@for\x:=#2\do{%
1136 \ifx\x\@empty
1137 \else
1138 \ifx\x#1\relax
1139 \else
1140 \edef\t{\the\toks@}%
1141 \ifx\t\@empty
1142 \else
1143 \toks@\expandafter{\the\toks@,}%
1144 \fi
1145 \toks@\expandafter{\the\expandafter\toks@\x}%
1146 \fi
1147 \fi
1148 }%
1149 \edef\x{\endgroup
1150 \def\noexpand#2{\the\toks@}%
1151 }%
1152 \x
1153 }
1154 \let\@@fileswith@pti@ns\KV0@fileswith@pti@ns

```

```

1155 \ifx\@fileswith@pti@ns\@badrequireerror
1156 \else
1157   \let\@fileswith@pti@ns\KV0@fileswith@pti@ns
1158 \fi

\KV0@Patch

1159 \let\KV0@Patch=Y

1160 \KV0@AtEnd
1161 </patch>

```

## 7 Test

### 7.1 Preface for standard catcode check

```

1162 <*test1>
1163 \input miniltx.tex\relax
1164 </test1>

```

### 7.2 Catcode checks for loading

```

1165 <*test1>

1166 \catcode'\@=11 %
1167 \def\RestoreCatcodes{}
1168 \count@=0 %
1169 \loop
1170   \edef\RestoreCatcodes{%
1171     \RestoreCatcodes
1172     \catcode\the\count@=\the\catcode\count@\relax
1173   }%
1174 \ifnum\count@<255 %
1175   \advance\count@\@ne
1176 \repeat
1177
1178 \def\RangeCatcodeInvalid#1#2{%
1179   \count@=#1\relax
1180   \loop
1181     \catcode\count@=15 %
1182     \ifnum\count@<#2\relax
1183       \advance\count@\@ne
1184     \repeat
1185 }
1186 \def\Test{%
1187   \RangeCatcodeInvalid{0}{47}%
1188   \RangeCatcodeInvalid{58}{64}%
1189   \RangeCatcodeInvalid{91}{96}%
1190   \RangeCatcodeInvalid{123}{255}%
1191   \catcode'\@=12 %
1192   \catcode'\=0 %
1193   \catcode'\{=1 %
1194   \catcode'\}=2 %
1195   \catcode'\#=6 %
1196   \catcode'\[=12 %
1197   \catcode'\]=12 %
1198   \catcode'\%=14 %
1199   \catcode'\ =10 %
1200   \catcode13=5 %
1201   \input kvoptions.sty\relax
1202   \RestoreCatcodes
1203 }
1204 \Test
1205 \csname @@end\endcsname
1206 \end

```



```

1207 </test1>

1208 <*test2>
1209 \NeedsTeXFormat{LaTeX2e}
1210 \makeatletter
1211 \catcode'\@=11 %
1212 \def\RestoreCatcodes{}
1213 \count@=0 %
1214 \loop
1215   \edef\RestoreCatcodes{%
1216     \RestoreCatcodes
1217     \catcode\the\count@=\the\catcode\count@\relax
1218   }%
1219 \ifnum\count@<255 %
1220   \advance\count@\@ne
1221 \repeat
1222
1223 \def\RangeCatcodeInvalid#1#2{%
1224   \count@=#1\relax
1225   \loop
1226     \catcode\count@=15 %
1227     \ifnum\count@<#2\relax
1228       \advance\count@\@ne
1229     \repeat
1230 }
1231 \def\Test#1{%
1232   \RangeCatcodeInvalid{0}{47}%
1233   \RangeCatcodeInvalid{58}{64}%
1234   \RangeCatcodeInvalid{91}{96}%
1235   \RangeCatcodeInvalid{123}{255}%
1236   \catcode'\@=12 %
1237   \catcode'\=0 %
1238   \catcode'\{=1 %
1239   \catcode'\}=2 %
1240   \catcode'\#=6 %
1241   \catcode'\[=12 %
1242   \catcode'\]=12 %
1243   \catcode'\%=14 %
1244   \catcode'\ =10 %
1245   \catcode13=5 %
1246   #1\relax
1247   \RestoreCatcodes
1248 }
1249 \Test{\RequirePackage{kvoptions-patch}}%
1250 \Test{\RequirePackage{kvoptions}}%
1251 \csname @@end\endcsname
1252 </test2>

```

## 8 Installation

### 8.1 Download

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

[CTAN:macros/latex/contrib/oberdiek/kvoptions.dtx](https://ctan.org/ctan/packages/macros/latex/contrib/oberdiek/kvoptions.dtx) The source file.

[CTAN:macros/latex/contrib/oberdiek/kvoptions.pdf](https://ctan.org/ctan/packages/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.

---

<sup>1</sup>[ftp://ftp.ctan.org/tex-archive/](http://ftp.ctan.org/tex-archive/)

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

*TDS* refers to the standard “A Directory Structure for T<sub>E</sub>X 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<sub>E</sub>X:

```
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-patch.sty</code>	→ <code>tex/latex/oberdiek/kvoptions-patch.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>test/kvoptions-test1.tex</code>	→ <code>doc/latex/oberdiek/test/kvoptions-test1.tex</code>
<code>test/kvoptions-test2.tex</code>	→ <code>doc/latex/oberdiek/test/kvoptions-test2.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`.

## 8.4 Refresh file name databases

If your T<sub>E</sub>X distribution (teT<sub>E</sub>X, mikT<sub>E</sub>X, ...) relies on file name databases, you must refresh these. For example, teT<sub>E</sub>X 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 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
```

## 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 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/](#)
- [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](#)

## 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<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 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-T<sub>E</sub>X added.
- Typos in documentation fixed (Axel Sommerfeldt).

[2007/10/11 v2.9]

- Bug fix for option patch.

[2007/10/18 v3.0]

- New package kvoptions-patch.

## 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> .....	1195, 1240
<code>\%</code> .....	1198, 1243
<code>\@</code> .....	1166, 1191, 1211, 1236
<code>\@@fileswith@pti@ns</code> .....	876, 1154
<code>\@unprocessedoptions</code> ...	1007, 1042
<code>\@SetupDriver</code> .....	41, 43
<code>\@badrequireerror</code> .....	1155
<code>\@classoptionslist</code> .....	573, 575, 818, 836, 922, 924
<code>\@cls@pkg</code> .....	983, 1018, 1033, 1036
<code>\@clsextension</code> .	253, 254, 273, 293, 294, 318, 331, 364, 392, 393, 414, 424, 445, 525, 537, 548, 571, 613, 815, 834, 892, 921, 1024
<code>\@current</code> .....	200, 211, 214, 217, 222, 225, 228, 253, 273, 274, 293, 364, 392, 432, 571, 607, 611, 613, 640, 682, 799, 805, 815, 834, 892, 963, 964, 969, 970, 973, 978, 1000, 1002, 1004, 1006, 1008, 1009, 1013, 1020, 1024, 1043, 1044, 1048
<code>\@currname</code> .....	57, 197, 211, 214, 215, 217, 222, 225, 226, 228, 252, 274, 292, 369, 391, 432, 607, 611, 640, 682, 799, 805, 962, 964, 1002, 1004, 1006, 1008, 1009, 1033, 1036, 1044, 1048
<code>\@curroptions</code> .....	800, 803, 820, 865, 1045, 1047, 1050
<code>\@declaredoptions</code> ....	811, 839, 872
<code>\@documentclasshook</code> .....	929
<code>\@ehc</code> .....	280, 460, 509, 545
<code>\@ehd</code> .....	757
<code>\@empty</code> 88, 89, 442, 491, 570, 623, 668, 676, 679, 689, 693, 798, 800, 812, 827, 837, 842, 875, 894, 952, 964, 965, 1002, 1045, 1051, 1059, 1079, 1080, 1093, 1136, 1141	
<code>\@expandtwoargs</code> .....	599
<code>\@fileswith@pti@ns</code> ...	876, 1155, 1157
<code>\@firstofone</code> .....	313, 316, 384
<code>\@firstoftwo</code> .....	1071
<code>\@for</code> .....	575, 616, 647, 678, 811, 836, 852, 865, 872, 912, 1050, 1135
<code>\@gobble</code> .....	327, 487, 703
<code>\@gobbletwo</code> .....	382
<code>\@if@pti@ns</code> .....	772
<code>\@if@ptions</code> .....	768, 970
<code>\@ifdefinable</code> .....	307
<code>\@ifl@aded</code> .....	969
<code>\@ifl@ter</code> .....	1013
<code>\@ifnextchar</code> .....	345
<code>\@ifpackageloaded</code> .....	760
<code>\@ifstar</code> .....	559, 703, 808
<code>\@ifundefined</code> .....	179, 182, 187, 190, 196, 199, 214, 225, 240, 267, 455, 576, 607, 617, 640, 648, 784, 799, 866, 973, 1044
<code>\@latex@error</code> .....	983, 1031
<code>\@latex@warning@no@line</code> .....	1015
<code>\@let@token</code> .....	1095, 1099
<code>\@missingfileerror</code> .....	1006
<code>\@namedef</code> .....	353, 403, 432
<code>\@nameuse</code> .....	682
<code>\@ne</code> .....	202, 1175, 1183, 1220, 1228
<code>\@nil</code> ..	577, 618, 649, 671, 681, 687, 694, 697, 942, 953, 1060, 1063, 1065, 1067, 1076, 1095, 1102, 1107, 1109, 1112, 1114, 1118, 1131
<code>\@onelevel@sanitize</code> .....	183, 311, 342, 343, 590, 621
<code>\@pass@ptions</code> .....	782, 1000
<code>\@pkgextension</code> .....	256, 296, 395, 449, 940, 947, 1043
<code>\@popfilename</code> .....	1025
<code>\@pushfilename</code> .....	961
<code>\@removeelement</code> .....	599, 882
<code>\@reset@ptions</code> .....	966, 1026
<code>\@secondoftwo</code> .....	1073
<code>\@tempc</code> .....	569
<code>\@twoloadclasserror</code> .....	1024
<code>\@typeset@protect</code> .....	194
<code>\@undefined</code> .....	1010
<code>\@unknownoptionerror</code> ..	68, 1030, 1051
<code>\@unprocessedoptions</code> .....	669, 877, 1007, 1011
<code>\@unusedoptionlist</code> .	600, 623, 624, 627, 629, 885, 888, 893, 894, 896
<code>\@x@protect</code> .....	191
<code>\[</code> .....	1196, 1241
<code>\[</code> .....	355, 1070, 1192, 1237
<code>\{</code> .....	1193, 1238

<code>\}</code> .....	1194, 1239	<code>\DisableKeyvalOption</code> .....	6, 451
<code>\]</code> .....	1197, 1242	<code>\do</code> .....	575, 616, 647, 678, 811, 836, 852, 865, 872, 912, 1050, 1135
<code>\_</code> .....	1199, 1244	<code>\documentclass</code> .....	756, 993
<code>\_</code> .....	1199, 1244	<code>\ds@</code> .....	798
<b>A</b>		<b>E</b>	
<code>\advance</code> .....	1175, 1183, 1220, 1228	<code>\emph</code> .....	48, 87, 91, 93
<code>\AtEndOfPackage</code> .....	204, 669, 877	<code>\empty</code> .....	109
<b>C</b>		<code>\end</code> .....	1206
<code>\catcode</code> .....	100, 101, 102, 103, 104, 128, 129, 130, 131, 132, 133, 134, 135, 153, 155, 159, 161, 202, 711, 713, 717, 719, 1166, 1172, 1181, 1191, 1192, 1193, 1194, 1195, 1196, 1197, 1198, 1199, 1200, 1211, 1217, 1226, 1236, 1237, 1238, 1239, 1240, 1241, 1242, 1243, 1244, 1245	<code>\endcsname</code> .....	105, 115, 136, 149, 152, 211, 217, 222, 228, 238, 239, 247, 285, 286, 287, 288, 307, 337, 374, 399, 462, 470, 474, 486, 490, 513, 518, 519, 521, 522, 611, 701, 710, 744, 774, 786, 791, 793, 805, 827, 842, 873, 889, 905, 913, 964, 978, 1002, 1008, 1009, 1020, 1048, 1205, 1251
<code>\ClassError</code> .....	538	<code>\endinput</code> .....	124, 739
<code>\ClassInfo</code> .....	332, 365, 425, 526	<code>\etex@unexpanded</code> .....	581, 585, 1085, 1092, 1121, 1122, 1126, 1127
<code>\ClassWarning</code> .....	319, 415, 549	<code>\ExecuteOptions</code> .....	903
<code>\count@</code> .....	1168, 1172, 1174, 1175, 1179, 1181, 1182, 1183, 1213, 1217, 1219, 1220, 1224, 1226, 1227, 1228	<b>F</b>	
<code>\csname</code> .....	105, 115, 136, 149, 152, 210, 217, 221, 228, 238, 239, 247, 285, 286, 287, 288, 307, 337, 374, 399, 462, 470, 474, 486, 490, 513, 518, 519, 521, 522, 611, 701, 710, 744, 774, 785, 790, 793, 805, 827, 842, 873, 889, 905, 913, 964, 978, 1002, 1008, 1009, 1020, 1048, 1205, 1251	<code>\futurelet</code> .....	1095
<code>\CurrentOption</code> .....	35, 37, 38, 41, 58, 62, 64, 408, 614, 616, 618, 621, 624, 630, 635, 645, 647, 649, 652, 656, 668, 678, 679, 681, 690, 694, 811, 812, 822, 827, 836, 837, 839, 842, 865, 872, 873, 875, 883, 898, 908, 912, 913, 918, 965, 1050, 1051, 1056	<b>G</b>	
<code>\CurrentOption@SaveLevel</code> .....	902, 905, 909, 910, 915, 916	<code>\gdef</code> .....	626, 785, 790, 924
<code>\CurrentOptionKey</code> .....	690, 693	<b>I</b>	
<code>\CurrentOptionValue</code> .....	56, 688, 689, 691, 698	<code>\ifcase</code> .....	106
<b>D</b>		<code>\ifetex@unexpanded</code> .....	751
<code>\DeclareBoolOption</code> .....	4, 20, 22, 234, 438	<code>\ifin@</code> .....	825, 840, 858
<code>\DeclareComplementaryOption</code> .....	5, 266, 439	<code>\ifKV@dyn@global</code> .....	468, 472, 479, 484, 488, 495, 516, 532
<code>\DeclareDefaultOption</code> .....	6, 55, 431	<code>\ifMCS@print</code> .....	79
<code>\DeclareOption</code> .....	202, 203	<code>\ifnum</code> .....	1174, 1182, 1219, 1227
<code>\DeclareStringOption</code> .....	4, 26, 28, 50, 52, 344, 440	<code>\ifpdf</code> .....	25
<code>\DeclareVoidOption</code> .....	5, 32, 33, 34, 380	<code>\ifx</code> .....	56, 88, 107, 109, 115, 136, 144, 253, 273, 293, 312, 315, 318, 331, 355, 364, 392, 412, 414, 424, 505, 525, 537, 548, 571, 573, 579, 613, 620, 623, 676, 679, 689, 701, 744, 812, 815, 834, 837, 853, 892, 894, 921, 922, 942, 944, 1024, 1043, 1051, 1070, 1080, 1099, 1136, 1138, 1141, 1155
<code>\default@ds</code> .....	868	<code>\immediate</code> .....	117, 138
<code>\define@key</code> .....	209, 220, 251, 291, 361, 390, 443, 447, 512	<code>\in@</code> .....	822
<code>\detokenize</code> .....	773, 777, 883, 885, 896, 898, 1070	<code>\in@false</code> .....	850
		<code>\in@true</code> .....	854, 859
		<code>\input</code> .....	1163, 1201
		<code>\InputIfFileExists</code> .....	1003
<b>K</b>		<b>K</b>	
		<code>\KV@sp@def</code> .....	698
		<code>\KV@setcurrentvalue</code> .....	697
		<code>\KV@action@error</code> .....	498
		<code>\KV@action@ignore</code> .....	482
		<code>\KV@action@undef</code> .....	466

\KVO@action@warning	501	\KVOdyn@ext	442, 445, 449, 514, 525
\KVO@AtEnd	157, 158, 706, 715, 716, 737, 738, 748, 758, 766, 1160	\KVOdyn@name	441, 444, 448, 505, 514, 530
\KVO@boolkey	252, 292, 309	<b>L</b>	
\KVO@calldefault	664, 672	\LoadClass	1024
\KVO@checkfirstA	1104, 1107	\loop	1169, 1180, 1214, 1225
\KVO@checkfirsttok	1095, 1098	<b>M</b>	
\KVO@checkkv	1065, 1076	\makeatletter	967, 1210
\KVO@CurrentOption	575, 577, 586, 590, 595, 599	\MCS@driver	84
\KVO@DeclareStringOption	346, 348, 351	\MCS@emph	88, 94
\KVO@disable@error	536	\meaning	184, 458, 978, 982, 1056
\KVO@disable@warning	547	\MessageBreak	58, 242, 243, 269, 274, 276, 278, 279, 324, 420, 458, 478, 494, 507, 531, 543, 554, 754, 762, 763, 985, 986, 988, 989, 990, 992, 994, 1017, 1018, 1019, 1020, 1036
\KVO@do@action	499, 502, 504	<b>N</b>	
\KVO@ExecuteOptions	904, 907	\NeedsTeXFormat	4, 709, 1209
\KVO@false	315, 340	\newcommand	40, 43, 231, 234, 266, 344, 380, 431, 451, 558
\KVO@family	209, 251, 275, 291, 361, 390, 562	\next	382, 384, 387
\KVO@fileswith@pti@ns	920, 1154, 1157	\numexpr	910, 916
\KVO@getkey	576, 618, 649, 671	<b>O</b>	
\KVO@ifdefinable	235, 236, 237, 283, 284, 306, 352, 383	\on@line	1016
\KVO@ifempty	1064, 1067, 1069, 1077, 1108, 1111, 1119	\OptionNotUsed	891
\KVO@in@	839, 849	<b>P</b>	
\KVO@nil	1095, 1118, 1122, 1131	\PackageError	268, 456, 506, 540, 753
\KVO@normalize	770, 783, 923, 1058	\PackageInfo	120, 334, 367, 427, 477, 493, 528
\KVO@onefilewithoptions	928, 933, 939, 946, 960	\PackageWarning	241, 321, 417, 551
\KVO@param	310, 311, 312, 315, 324, 336, 337	\PackageWarningNoLine	57, 745, 761
\KVO@Patch	579, 620, 1159	\PassOptionsToPackage	63, 80
\KVO@prefix	220, 235, 236, 237, 239, 240, 247, 258, 267, 277, 283, 284, 285, 286, 287, 288, 298, 352, 353, 374, 383, 399, 403	\ProcessKeyvalOptions	3, 74, 558, 702
\KVO@process@pti@ns	831, 847, 864	\ProcessOptions	208, 797
\KVO@process@options	808, 810	\protect	192
\KVO@ProcessKeyvalOptions	562, 566, 568	\ProvidesPackage	5, 150, 741
\KVO@removeelement	1132	<b>R</b>	
\KVO@removegarbage	1122, 1131	\RangeCatcodeInvalid	1178, 1187, 1188, 1189, 1190, 1223, 1232, 1233, 1234, 1235
\KVO@removelastspace	1101, 1109, 1112, 1114, 1118	\renewcommand	93
\KVO@result	1059, 1061, 1084, 1085, 1091, 1092, 1120, 1121, 1125, 1126	\repeat	1176, 1184, 1221, 1229
\KVO@SanitizedCurrentOption	866, 889, 1032, 1035, 1055	\RequirePackage	7, 8, 84, 180, 205, 750, 755, 1249, 1250
\KVO@setcurrents	681, 687	\reserved@a	927, 932, 952, 953, 956, 958, 968, 1028
\KVO@setcurrentvalue	694, 697	\reserved@b	941, 949, 953
\KVO@splitcomma	1060, 1063, 1067	\RestoreCatcodes	1167, 1170, 1171, 1202, 1212, 1215, 1216, 1247
\KVO@temp	570, 580, 582, 592, 593, 608, 615, 642, 646, 663, 667, 770, 777, 783, 787, 793, 923, 925, 938, 946, 955	<b>S</b>	
\KVO@true	312, 340	\setkeys	44, 232, 453, 665
\KVO@use@option	826, 841, 867, 870, 879	\SetupDriver	32, 33, 34, 35, 38, 40
\KVO@VOID@	390, 406, 412	\SetupKeyvalOptions	3, 13, 231, 434
\KVO@voidkey	391, 407	\space	273, 983, 986, 989, 991, 995, 1018, 1033, 1036, 1038, 1039
\KVO@x	1079, 1080, 1085	\strip@prefix	184, 188, 458, 977, 982, 1056
\KVO@xprocess@options	808, 833		
\KVOdyn@action	455, 458, 462		

T		U	
\t	1140, 1141	\unexpanded	754
\Test	1186, 1204, 1231, 1249, 1250		
\textcolor	94		
		W	
\the	153, 159, 361, 370, 614, 635, 642, 645, 652, 656, 664, 665, 711, 717, 822, 910, 916, 955, 1140, 1143, 1145, 1150, 1172, 1217	\write	117, 138
		X	
\TMP@EnsureCode	156, 163, 164, 165, 166, 167, 168, 169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 714, 721, 722, 723, 724, 725, 726, 727, 728, 729, 730, 731, 732, 733, 734, 735, 736	\x	105, 107, 109, 116, 120, 122, 137, 142, 149, 250, 261, 290, 301, 360, 377, 389, 402, 410, 412, 454, 464, 467, 483, 511, 561, 564, 674, 676, 771, 780, 821, 824, 852, 853, 857, 862, 881, 888, 974, 976, 986, 992, 1135, 1136, 1138, 1145, 1149, 1152
\toks	606, 651, 652, 664, 820, 822		
		Y	
\toks@	356, 358, 361, 363, 370, 608, 610, 614, 615, 634, 635, 641, 642, 645, 646, 655, 656, 665, 816, 818, 822, 954, 955, 1134, 1140, 1143, 1145, 1150	\y	981, 982, 989, 992
		Z	
\tw@	606, 651, 652, 664, 820, 822	\zap@space	693, 952, 1079, 1093