

# 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	24
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>34</b>
8.1	Download	34
8.2	Bundle installation	35
8.3	Package installation	35
8.4	Refresh file name databases	35
8.5	Some details for the interested	35
<b>9</b>	<b>References</b>	<b>36</b>
<b>10</b>	<b>History</b>	<b>36</b>
	[0000/00/00 v0.0]	36
	[2004/02/22 v1.0]	37
	[2006/02/16 v2.0]	37
	[2006/02/20 v2.1]	37
	[2006/06/01 v2.2]	37
	[2006/08/17 v2.3]	37
	[2006/08/22 v2.4]	37
	[2007/04/11 v2.5]	37
	[2007/05/06 v2.6]	37
	[2007/06/11 v2.7]	37
	[2007/10/02 v2.8]	37
	[2007/10/11 v2.9]	38
	[2007/10/18 v3.0]	38
<b>11</b>	<b>Index</b>	<b>38</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

`\DeclareDefaultOption{⟨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

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

`⟨options⟩:`

<code>action</code>	<code>= undef, warning, error, or ignore</code>	<code>default: undef</code>
<code>global or local</code>		<code>default: 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  $\text{\LaTeX}$ 's kernel commands in order to get better support for key value options. Of course they cannot be used both. The user must decide, which method he prefers. Package `kvoptions-patch` aborts itself, if it detects that `xkvltxp` is already loaded.

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

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

The other way should work, too.

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

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

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

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

### 5.2 Limitations

#### 5.2.1 Option comparisons

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

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

### 5.2.2 Option list parsing with package `kvoptions-patch`

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

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

Example:

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

Result:

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

## 6 Implementation

### 6.1 Preamble

```
98 <*package>
```

**Reload check and identification.** Reload check, especially if the package is not used with L<sup>A</sup>T<sub>E</sub>X.

```
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   \catcode35 6 % #
116   \catcode123 1 % {
117   \catcode125 2 % }
118   \expandafter\ifx\csname PackageInfo\endcsname\relax
119     \def\x#1#2{%
120       \immediate\write-1{Package #1 Info: #2.}%
121     }%
122   \else
123     \def\x#1#2{\PackageInfo{#1}{#2, stopped}}%
124   \fi
125   \x{kvoptions}{The package is already loaded}%
126 \endgroup
127 \expandafter\endinput
128 \fi
129 \endgroup
```

Package identification:

```
130 \begingroup
131   \catcode35 6 % #
132   \catcode40 12 % (
133   \catcode41 12 % )
134   \catcode44 12 % ,
135   \catcode45 12 % -
136   \catcode46 12 % .
137   \catcode47 12 % /
138   \catcode58 12 % :
139   \catcode64 11 % @
140   \catcode123 1 % {
141   \catcode125 2 % }
142   \expandafter\ifx\csname ProvidesPackage\endcsname\relax
143     \def\x#1#2#3[#4]{\endgroup
144       \immediate\write-1{Package: #3 #4}%
145       \xdef#1{#4}%
146     }%
147   \else
148     \def\x#1#2[#3]{\endgroup
149       #2[#{#3}]%
150       \ifx#1\relax
151         \xdef#1{#3}%
152       \fi
153     }%
154   \fi
155   \expandafter\x\csname ver@kvoptions.sty\endcsname
156   \ProvidesPackage{kvoptions}%
157   [2007/10/18 v3.0 Keyval support for LaTeX options (H0)]
```

## Catcodes

```
158 \begingroup
159   \catcode123 1 % {
160   \catcode125 2 % }
161   \def\x{\endgroup
162     \expandafter\edef\csname KVO@AtEnd\endcsname{%
163       \catcode35 \the\catcode35\relax
164       \catcode64 \the\catcode64\relax
165       \catcode123 \the\catcode123\relax
166       \catcode125 \the\catcode125\relax
167     }%
168   }%
169   \x
170   \catcode35 6 % #
171   \catcode64 11 % @
172   \catcode123 1 % {
173   \catcode125 2 % }
174   \def\TMP@EnsureCode#1#2{%
175     \edef\KVO@AtEnd{%
176       \KVO@AtEnd
177       \catcode#1 \the\catcode#1\relax
178     }%
179     \catcode#1 #2\relax
180   }
181   \TMP@EnsureCode{1}{14}% ^^A (comment)
182   \TMP@EnsureCode{2}{14}% ^^A (comment)
183   \TMP@EnsureCode{33}{12}% !
184   \TMP@EnsureCode{39}{12}% '
185   \TMP@EnsureCode{40}{12}% (
186   \TMP@EnsureCode{41}{12}% )
187   \TMP@EnsureCode{42}{12}% *
188   \TMP@EnsureCode{44}{12}% ,
```

```

189 \TMP@EnsureCode{45}{12}% -
190 \TMP@EnsureCode{46}{12}% .
191 \TMP@EnsureCode{47}{12}% /
192 \TMP@EnsureCode{58}{12}% :
193 \TMP@EnsureCode{61}{12}% =
194 \TMP@EnsureCode{62}{12}% >
195 \TMP@EnsureCode{94}{7}% ^ (superscript)
196 \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`.

```

197 \@ifundefined{define@key}{%
198   \RequirePackage{keyval}\relax
199 }{}

```

### Provide macros for plain-TeX.

```

200 \@ifundefined{@onelevel@sanitize}{%
201   \def\@onelevel@sanitize#1{%
202     \edef#1{\expandafter\strip@prefix\meaning#1}%
203   }%
204 }{}
205 \@ifundefined{strip@prefix}{%
206   \def\strip@prefix#1>{%
207   }%
208 }{}
209 \@ifundefined{@x@protect}{%
210   \def\@x@protect#1\fi#2#3{%
211     \fi\protect#1%
212   }%
213 }{}
214 \@ifundefined{@currname}{%
215   \def\@currname{}%
216 }{}
217 \@ifundefined{@currentx}{%
218   \def\@currentx{}%
219 }{}

```

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

```

220 \DeclareOption{debugshow}{\catcode\@ne=9 }
221 \DeclareOption{patch}{%
222   \AtEndOfPackage{%
223     \RequirePackage{kvoptions-patch}[2007/10/18]%
224   }%
225 }

```

Optionen auswerten:

```

226 \ProcessOptions\relax

```

## 6.2 Option declaration macros

### 6.2.1 \SetupKeyvalOptions

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

`\KV0@family` We cannot store the family setting in one macro, because the package should be usable for many other packages, too. Thus we remember the family setting in a macro, whose name contains the package name with extension, a key in L<sup>A</sup>T<sub>E</sub>X's class/package system.

```

227 \define@key{KV0}{family}{%
228   \expandafter\edef\csname KV0@family@%
229     \@currname.\@current\endcsname{#1}%
230 }
231 \def\KV0@family{%
232   \ifundefined{KV0@family@\@currname.\@current}{%
233     \@currname
234   }{%
235     \csname KV0@family@\@currname.\@current\endcsname
236   }%
237 }

```

`\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.

```

238 \define@key{KV0}{prefix}{%
239   \expandafter\edef\csname KV0@prefix@%
240     \@currname.\@current\endcsname{#1}%
241 }
242 \def\KV0@prefix{%
243   \ifundefined{KV0@prefix@\@currname.\@current}{%
244     \@currname @%
245   }{%
246     \csname KV0@prefix@\@currname.\@current\endcsname
247   }%
248 }

```

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

```

249 \newcommand*\SetupKeyvalOptions{%
250   \setkeys{KV0}%
251 }

```

## 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.

```

252 \newcommand*\DeclareBoolOption}[2][false]{%
253   \KV0@ifdefinable{if\KV0@prefix#2}{%
254     \KV0@ifdefinable{\KV0@prefix#2true}{%
255       \KV0@ifdefinable{\KV0@prefix#2false}{%
256         \csname newif\expandafter\endcsname
257         \csname if\KV0@prefix#2\endcsname
258       }%
259       \ifundefined{KV0@prefix#2#1}{%
260         \PackageWarning{kvoptions}{%
261           Initialization of option '#2' failed,\MessageBreak
262           cannot set boolean option to '#1',\MessageBreak
263           use 'true' or 'false', now using 'false'%

```

```

263     }%
264   }{%
265     \csname\KV0@prefix#2#1\endcsname
266   }%
267   \begingroup
268     \edef\x{\endgroup
269       \noexpand\define@key{\KV0@family}{#2}[true]{%
270         \noexpand\KV0@boolkey{\@currname}%
271         \ifx\@current\@clsextension
272           \noexpand\@clsextension
273         \else
274           \noexpand\@pkgextension
275         \fi
276         {\KV0@prefix}{#2}{####1}%
277       }%
278     }%
279   \x
280 }%
281 }%
282 }%
283 }

```

`\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.

```

284 \newcommand*{\DeclareComplementaryOption}[2]{%
285   \@ifundefined{if\KV0@prefix#2}{%
286     \PackageError{kvoptions}{%
287       Cannot generate option code for ‘#1’,\MessageBreak
288       parent switch ‘#2’ does not exist%
289     }{%
290       You are inside %
291       \ifx\@current\@clsextension class\else package\fi\space
292       ‘\@currname.\@current’.\MessageBreak
293       ‘\KV0@family’ is used as family %
294       for the keyval options.\MessageBreak
295       ‘\KV0@prefix’ serves as prefix %
296       for internal switch macros.\MessageBreak
297       \MessageBreak
298       \@ehc
299     }%
300   }{%
301     \KV0@ifdefinable{\KV0@prefix#1true}{%
302       \KV0@ifdefinable{\KV0@prefix#1false}{%
303         \expandafter\let\csname\KV0@prefix#1false\expandafter\endcsname
304         \csname\KV0@prefix#2true\endcsname
305         \expandafter\let\csname\KV0@prefix#1true\expandafter\endcsname
306         \csname\KV0@prefix#2false\endcsname

```

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

```

307   \begingroup
308     \edef\x{\endgroup
309       \noexpand\define@key{\KV0@family}{#1}[true]{%
310         \noexpand\KV0@boolkey{\@currname}%
311         \ifx\@current\@clsextension
312           \noexpand\@clsextension
313         \else
314           \noexpand\@pkgextension
315         \fi
316         {\KV0@prefix}{#1}{####1}%
317       }%
318     }%

```



```

319         \x
320     }%
321 }%
322 }%
323 }

```

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

```

324 \def\KV0@ifdefinable#1{%
325     \expandafter\@ifdefinable\csname #1\endcsname
326 }

```

`\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

```

```

327 \def\KV0@boolkey#1#2#3#4#5{%
328     \edef\KV0@param{#5}%
329     \@onelevel@sanitize\KV0@param
330     \ifx\KV0@param\KV0@true
331         \expandafter\@firstofone
332     \else
333         \ifx\KV0@param\KV0@false
334             \expandafter\expandafter\expandafter\@firstofone
335         \else
336             \ifx#2\@clsextension
337                 \expandafter\ClassWarning
338             \else
339                 \expandafter\PackageWarning
340             \fi
341             {#1}{%
342                 Value ‘\KV0@param’ is not supported by\MessageBreak
343                 option ‘#4’%
344             }%
345             \expandafter\expandafter\expandafter\@gobble
346         \fi
347     \fi
348     {%
349         ^^A\ifx#2\@clsextension
350         ^^A \expandafter\ClassInfo
351         ^^A\else
352         ^^A \expandafter\PackageInfo
353         ^^A\fi
354         ^^A{#1}{[option] #4=\KV0@param}%
355         \csname#3#4\KV0@param\endcsname
356     }%
357 }

```

`\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).

```

358 \def\KV0@true{true}
359 \def\KV0@false{false}
360 \@onelevel@sanitize\KV0@true
361 \@onelevel@sanitize\KV0@false

```

### 6.2.3 `\DeclareStringOption`

`\DeclareStringOption`

```

362 \newcommand*{\DeclareStringOption}[2][{}]{%
363   \@ifnextchar[%
364     \KV0@DeclareStringOption{#1}{#2}@%
365   }{%
366     \KV0@DeclareStringOption{#1}{#2}{}[{}]{%
367   }%
368 }

```

\KV0@DeclareStringOption

```

369 \def\KV0@DeclareStringOption#1#2#3[#4]{%
370   \KV0@ifdefinable{\KV0@prefix#2}{%
371     \@namedef{\KV0@prefix#2}{#1}%
372     \begingroup
373       \ifx\#3\%
374         \toks@{%
375       \else
376         \toks@{[#4]}%
377       \fi
378       \edef\x{\endgroup
379         \noexpand\define@key{\KV0@family}{#2}\the\toks@{%
380         ^^A\begingroup
381         ^^A \toks@{####1}%
382         ^^A \ifx\@current\@clsextension
383         ^^A \noexpand\ClassInfo
384         ^^A \else
385         ^^A \noexpand\PackageInfo
386         ^^A \fi
387         ^^A {\@currname}{%
388         ^^A [option] #2={\noexpand\the\toks@}%
389         ^^A }%
390         ^^A\endgroup
391       \noexpand\def
392       \expandafter\noexpand\csname\KV0@prefix#2\endcsname{####1}%
393     }%
394   }%
395   \x
396 }%
397 }

```

## 6.2.4 \DeclareVoidOption

\DeclareVoidOption

```

398 \newcommand*{\DeclareVoidOption}[1]{%
399   \begingroup
400     \let\next\@gobbletwo
401     \KV0@ifdefinable{\KV0@prefix#1}{%
402       \let\next\@firstofone
403     }%
404   \expandafter\endgroup
405   \next{%
406     \begingroup
407     \edef\x{\endgroup
408       \noexpand\define@key{\KV0@family}{#1}{\KV0@VOID@}{%
409       \noexpand\KV0@voidkey{\@currname}%
410       \ifx\@current\@clsextension
411         \noexpand\@clsextension
412       \else
413         \noexpand\@pkgextension
414       \fi
415       {#1}%
416       {####1}%
417       \expandafter\noexpand\csname\KV0@prefix#1\endcsname

```

```

418         }%
419     }%
420     \x
421     \@namedef{\KV0@prefix#1}%
422 }%
423 }
424 \def\KV0@VOID@{@VOID@}

#1 package/class name
#2 \@pkgextension/\@clsextension
\KV0@voidkey #3 key name
#4 default (@VOID@)
#5 macro with option code

425 \def\KV0@voidkey#1#2#3#4{%
426     \def\CurrentOption{#3}%
427     \begingroup
428         \def\x{#4}%
429     \expandafter\endgroup
430     \ifx\x\KV0@VOID@
431     \else
432         \ifx#2\@clsextension
433             \expandafter\ClassWarning
434         \else
435             \expandafter\PackageWarning
436         \fi
437         {#1}{%
438             Unexpected value for option ‘#3’\MessageBreak
439             is ignored%
440         }%
441     \fi
442     ^^A\ifx#2\@clsextension
443     ^^A \expandafter\ClassInfo
444     ^^A\else
445     ^^A \expandafter\PackageInfo
446     ^^A\fi
447     ^^A{#1}{[option] #3}%
448 }

```

### 6.2.5 \DeclareDefaultOption

\DeclareDefaultOption

```

449 \newcommand*{\DeclareDefaultOption}{%
450     \@namedef{KV0@default@\currname.\currentx}%
451 }

```

## 6.3 Dynamic options

### 6.3.1 \DisableKeyvalOption

```

452 \SetupKeyvalOptions{%
453     family=KV0dyn,%
454     prefix=KV0dyn%
455 }
456 \DeclareBoolOption[true]{global}
457 \DeclareComplementaryOption{local}{global}
458 \DeclareStringOption[undef]{action}
459 \let\KV0dyn@name\relax
460 \let\KV0dyn@ext\@empty
461 \define@key{KV0dyn}{class}{%
462     \def\KV0dyn@name{#1}%
463     \let\KV0dyn@ext\@clsextension
464 }

```

```

465 \define@key{KV0dyn}{package}{%
466   \def\KV0dyn@name{#1}%
467   \let\KV0dyn@ext\@pkgextension
468 }
469 \newcommand*{\DisableKeyvalOption}[3][ ]{%
470   \begin{group}
471     \setkeys{KV0dyn}{#1}%
472     \def\x{\endgroup}%
473     \ifundefined{KV0@action@\KV0dyn@action}{%
474       \PackageError{kvoptions}{%
475         Unknown disable action %
476         '\expandafter\strip@prefix\meaning\KV0dyn@action'\MessageBreak
477         for option '#3' in keyval family '#2'%
478       } \@ehc
479     }{%
480       \csname KV0@action@\KV0dyn@action\endcsname{#2}{#3}%
481     }%
482   \x
483 }
484 \def\KV0@action@undef#1#2{%
485   \edef\x{\endgroup}
486   \ifKV0dyn@global\global\fi
487   \let
488   \expandafter\noexpand\csname KV@#1@#2\endcsname
489   \relax
490   \ifKV0dyn@global\global\fi
491   \let
492   \expandafter\noexpand\csname KV@#1@#2@default\endcsname
493   \relax
494 }%
495 ^^A\PackageInfo{kvoptions}{%
496   ^^A [option] key '#2' of family '#1'\MessageBreak
497   ^^A is disabled (undef, \ifKV0dyn@global\global\else local\fi)%
498   ^^A}%
499 }
500 \def\KV0@action@ignore#1#2{%
501   \edef\x{\endgroup}
502   \ifKV0dyn@global\global\fi
503   \let
504   \expandafter\noexpand\csname KV@#1@#2\endcsname
505   \@gobble
506   \ifKV0dyn@global\global\fi
507   \let
508   \expandafter\noexpand\csname KV@#1@#2@default\endcsname
509   \@empty
510 }%
511 ^^A\PackageInfo{kvoptions}{%
512   ^^A [option] key '#2' of family '#1'\MessageBreak
513   ^^A is disabled (ignore, \ifKV0dyn@global\global\else local\fi)%
514   ^^A}%
515 }
516 \def\KV0@action@error{%
517   \KV0@do@action{error}%
518 }
519 \def\KV0@action@warning{%
520   \KV0@do@action{warning}%
521 }
#1 error or warning
#2 <family>
#3 <key>
522 \def\KV0@do@action#1#2#3{%
523   \ifx\KV0dyn@name\relax

```

```

524 \PackageError{kvoptions}{%
525   Action type '#1' needs package/class name\MessageBreak
526   for key '#3' in family '#2'%
527 } \@ehc
528 \else
529 \edef\x{\endgroup
530   \noexpand\define@key{#2}{#3}[] {%
531     \expandafter\noexpand\csname KV0@disable@#1\endcsname
532     {\KV0dyn@name}\noexpand\KV0dyn@ext{#3}%
533   }%
534   \ifKV0dyn@global
535     \global\let
536     \expandafter\noexpand\csname KV@#2@#3\endcsname
537     \expandafter\noexpand\csname KV@#2@#3\endcsname
538     \global\let
539     \expandafter\noexpand\csname KV@#2@#3@default\endcsname
540     \expandafter\noexpand\csname KV@#2@#3@default\endcsname
541   \fi
542 }%
543 ^^A\ifx\KV0dyn@ext\@clsextension
544 ^^A \expandafter\ClassInfo
545 ^^A\else
546 ^^A \expandafter\PackageInfo
547 ^^A\fi
548 ^^A{\KV0dyn@name}{%
549 ^^A [option] key '#3' of family '#2'\MessageBreak
550 ^^A is disabled (#1, \ifKV0dyn@global global\else local\fi)%
551 ^^A}%
552 \fi
553 }
554 \def\KV0@disable@error#1#2#3{%
555   \ifx#2\@clsextension
556     \expandafter\ClassError
557   \else
558     \expandafter\PackageError
559   \fi
560   {#1}{%
561     Option '#3' is given too late,\MessageBreak
562     now the option is ignored%
563   } \@ehc
564 }
565 \def\KV0@disable@warning#1#2#3{%
566   \ifx#2\@clsextension
567     \expandafter\ClassWarning
568   \else
569     \expandafter\PackageWarning
570   \fi
571   {#1}{%
572     Option '#3' is already consumed\MessageBreak
573     and has no effect%
574   }%
575 }

```

## 6.4 Process options

### 6.5 \ProcessKeyvalOptions

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

```

576 \newcommand*{\ProcessKeyvalOptions}{%
577   \@ifstar{%
578     \begingroup
579     \edef\x{\endgroup
580       \noexpand\KV0@ProcessKeyvalOptions{\KV0@family}%

```

```

581     }%
582     \x
583 }%
584 \KVO@ProcessKeyvalOptions
585 }

```

```

586 \def\KVO@ProcessKeyvalOptions#1{%
587   \let\@tempc\relax
588   \let\KVO@temp\@empty

```

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

```

589   \ifx\@currentx\@clsextension
590   \else
591     \ifx\@classoptionslist\relax
592     \else
593       \@for\KVO@CurrentOption:=\@classoptionslist\do{%
594         \ifundefined{KV@#1}\expandafter\KVO@getkey
595           \KVO@CurrentOption=\@nil}{%
596       }{%
597         \ifx\KVO@Patch Y%
598           \edef\KVO@temp{%
599             \etex@unexpanded\expandafter{%
600               \KVO@temp
601             }%
602             ,%
603             \etex@unexpanded\expandafter{%
604               \KVO@CurrentOption
605             }%
606             ,%
607           }%
608           \@onelevel@sanitize\KVO@CurrentOption
609         \else
610           \edef\KVO@temp{%
611             \KVO@temp
612             ,%
613             \KVO@CurrentOption
614             ,%
615           }%
616         \fi
617         \@expandtwoargs\@removeelement\KVO@CurrentOption
618         \@unusedoptionlist\@unusedoptionlist
619       }%
620     }%
621   \fi
622 \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.

```

623 \begingroup
624   \toks\tw@{}%
625   \@ifundefined{opt@\@currname.\@currentx}{%
626     \toks@\expandafter{\KVO@temp}%
627   }{%
628     \toks@\expandafter\expandafter\expandafter{%
629       \csname opt@\@currname.\@currentx\endcsname
630     }%
631     \ifx\@currentx\@clsextension
632       \edef\CurrentOption{\the\toks@}%
633       \toks@\expandafter{\KVO@temp}%
634       \@for\CurrentOption:=\CurrentOption\do{%
635         \@ifundefined{%

```

```

636         KV@#1@\expandafter\KV0@getkey\CurrentOption=\@nil
637     }{%
A class puts not used options in the unused option list.
638         \ifx\KV0@Patch Y%
639             \@onelevel@sanitize\CurrentOption
640         \fi
641         \ifx\@unusedoptionlist\@empty
642             \global\let\@unusedoptionlist\CurrentOption
643         \else
644             \expandafter\expandafter\expandafter\gdef
645             \expandafter\expandafter\expandafter\@unusedoptionlist
646             \expandafter\expandafter\expandafter{%
647                 \expandafter\@unusedoptionlist
648                 \expandafter,\CurrentOption
649             }%
650         \fi
651     }{%
652         \toks@\expandafter{%
653             \the\expandafter\toks@\expandafter,\CurrentOption
654         }%
655     }%
656 }%
657 \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.

```

658     \@ifundefined{KV0@default@\@currname.\@currentx}{%
659         \toks@\expandafter\expandafter\expandafter{%
660             \expandafter\KV0@temp\the\toks@
661         }%
662     }{%
663         \edef\CurrentOption{\the\toks@}%
664         \toks@\expandafter{\KV0@temp}%
665         \@for\CurrentOption:=\CurrentOption\do{%
666             \@ifundefined{%
667                 KV@#1@\expandafter\KV0@getkey\CurrentOption=\@nil
668             }{%
669                 \toks\tw@\expandafter{%
670                     \the\toks@\expandafter\tw@\expandafter,\CurrentOption
671                 }%
672             }{%
673                 \toks@\expandafter{%
674                     \the\expandafter\toks@\expandafter,\CurrentOption
675                 }%
676             }%
677         }%
678     }%
679 \fi
680 }%
681 \edef\KV0@temp{\endgroup
682     \noexpand\KV0@calldefault{\the\toks\tw@}%
683     \noexpand\setkeys{#1}{\the\toks@}%
684 }%
685 \KV0@temp

```

Some cleanup of `\ProcessOptions`.

```

686 \let\CurrentOption\@empty
687 \AtEndOfPackage{\let\@unprocessedoptions\relax}%
688 }

```

### 6.5.1 Helper macros

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

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

`\KV0@calldefault`

```
690 \def\KV0@calldefault#1{%
691   \begingroup
692   \def\x{#1}%
693   \expandafter\endgroup
694   \ifx\x\@empty
695   \else
696     \@for\CurrentOption:=#1\do{%
697       \ifx\CurrentOption\@empty
698       \else
699         \expandafter\KV0@setcurrents\CurrentOption=\@nil
700         \@nameuse{KV0@default@\@currname.\@current}%
701       \fi
702     }%
703   \fi
704 }
```

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

```
705 \def\KV0@setcurrents#1=#2\@nil{
706   \def\CurrentValue{#2}%
707   \ifx\CurrentValue\@empty
708     \let\CurrentOptionKey\CurrentOption
709     \let\CurrentValue\relax
710   \else
711     \edef\CurrentOptionKey{\zap@space#1 \@empty}%
712     \expandafter\KV0@setcurrentvalue\CurrentOption\@nil
713   \fi
714 }
```

`\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.

```
715 \def\KV0@setcurrentvalue#1=#2\@nil{%
716   \KV@@sp@def\CurrentValue{#2}%
717 }
```

## 6.6 plain-TeX

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

```
718 \begingroup\expandafter\expandafter\expandafter\endgroup
719 \expandafter\ifx\csname documentclass\endcsname\relax
720   \def\ProcessKeyvalOptions{%
721     \@ifstar{}\@gobble
722   }%
723 \fi
724 \KV0@AtEnd
725 \endpackage
```

## 6.7 Package `kvoptions-patch`

```
726 \patch{
727   \NeedsTeXFormat{LaTeX2e}
728   \begingroup
729     \catcode123 1 % {
730     \catcode125 2 % }
731   \def\x{\endgroup
732     \expandafter\edef\csname KV0@AtEnd\endcsname{%
```



```

733     \catcode35 \the\catcode35\relax
734     \catcode64 \the\catcode64\relax
735     \catcode123 \the\catcode123\relax
736     \catcode125 \the\catcode125\relax
737   }%
738 }%
739 \x
740 \catcode35 6 % #
741 \catcode64 11 % @
742 \catcode123 1 % {
743 \catcode125 2 % }
744 \def\TMP@EnsureCode#1#2{%
745   \edef\KVO@AtEnd{%
746     \KVO@AtEnd
747     \catcode#1 \the\catcode#1\relax
748   }%
749   \catcode#1 #2\relax
750 }
751 \TMP@EnsureCode{39}{12}% '
752 \TMP@EnsureCode{40}{12}% (
753 \TMP@EnsureCode{41}{12}% )
754 \TMP@EnsureCode{43}{12}% +
755 \TMP@EnsureCode{44}{12}% ,
756 \TMP@EnsureCode{45}{12}% -
757 \TMP@EnsureCode{46}{12}% .
758 \TMP@EnsureCode{47}{12}% /
759 \TMP@EnsureCode{58}{12}% :
760 \TMP@EnsureCode{60}{12}% <
761 \TMP@EnsureCode{61}{12}% =
762 \TMP@EnsureCode{62}{12}% >
763 \TMP@EnsureCode{91}{12}% [
764 \TMP@EnsureCode{93}{12}% ]
765 \TMP@EnsureCode{96}{12}% `
766 \TMP@EnsureCode{124}{12}% |
767 \edef\KVO@AtEnd{%
768   \KVO@AtEnd
769   \noexpand\endinput
770 }
771 \ProvidesPackage{kvoptions-patch}%
772 [2007/10/18 v3.0 LaTeX patch for keyval options (H0)]%

  Check for  $\varepsilon$ -TeX.
773 \begingroup\expandafter\expandafter\expandafter\endgroup
774 \expandafter\ifx\csname eTeXversion\endcsname\relax
775   \PackageWarningNoLine{kvoptions-patch}{%
776     Package loading is aborted, because e-TeX is missing%
777   }%
778   \expandafter\KVO@AtEnd
779 \fi

  Package etexcmds for \etex@unexpanded.
780 \RequirePackage{etexcmds}[2007/09/09]
781 \ifetex@unexpanded
782 \else
783   \PackageError{kvoptions-patch}{%
784     Could not find eTeX's \string\unexpanded.\MessageBreak
785     Try adding \string\RequirePackage\string{etexcmds\string} %
786     before \string\documentclass%
787   }\@ehd
788   \expandafter\KVO@AtEnd
789 \fi

  Check for package xkvltxp.
790 \@ifpackageloaded{xkvltxp}{%

```

```

791 \PackageWarningNoLine{kvoptions}{%
792   Option 'patch' cannot be used together with\MessageBreak
793   package 'xkvltxp' that is already loaded.\MessageBreak
794   Therefore package loading is aborted%
795 }%
796 \KV0@AtEnd
797 }{}

798 \def\@if@options#1#2#3{%
799   \begingroup
800     \KV0@normalize\KV0@temp{#3}%
801     \edef\x{\endgroup
802       \noexpand\@if@ptions{%
803         \detokenize\expandafter\expandafter\expandafter{%
804           \csname opt@#2.#1\endcsname
805         }%
806       }{%
807         \detokenize\expandafter{\KV0@temp}%
808       }%
809     }%
810     \x
811 }

812 \def\@pass@options#1#2#3{%
813   \KV0@normalize\KV0@temp{#2}%
814   \@ifundefined{opt@#3.#1}{%
815     \expandafter\gdef\csname opt@#3.#1%
816       \expandafter\endcsname\expandafter{%
817         \KV0@temp
818       }%
819   }{%
820     \expandafter\gdef\csname opt@#3.#1%
821       \expandafter\expandafter\expandafter\endcsname
822       \expandafter\expandafter\expandafter{%
823         \csname opt@#3.#1\expandafter\endcsname\expandafter,\KV0@temp
824       }%
825   }%
826 }

827 \def\ProcessOptions{%
828   \let\ds@\@empty
829   \@ifundefined{opt@\@currname.\@currentx}{%
830     \let\@curroptions\@empty
831   }{%
832     \expandafter\expandafter\expandafter\def
833     \expandafter\expandafter\expandafter\@curroptions
834     \expandafter\expandafter\expandafter{%
835       \csname opt@\@currname.\@currentx\endcsname
836     }%
837   }%
838   \@ifstar\KV0@xprocessoptions\KV0@processoptions
839 }

840 \def\KV0@processoptions{%
841   \@for\CurrentOption:=\@declaredoptions\do{%
842     \ifx\CurrentOption\@empty
843     \else
844       \begingroup
845         \ifx\@currentx\@clsextension
846           \toks@{}%
847         \else
848           \toks@\expandafter{\@classoptionslist,%
849         \fi
850         \toks\@tw@\expandafter{\@curroptions}%
851         \edef\x{\endgroup

```

```

852         \noexpand\in@{,\CurrentOption,}{,\the\toks@\the\toks\tw@,}%
853     }%
854     \x
855     \ifin@
856         \KV0@use@ption
857         \expandafter\let\csname ds@\CurrentOption\endcsname\@empty
858     \fi
859     \fi
860 }%
861 \KV0@process@ptions
862 }

863 \def\KV0@xprocess@ptions{%
864     \ifx\@currentx\@clsextension
865     \else
866         \@for\CurrentOption:=\@classoptionslist\do{%
867             \ifx\CurrentOption\@empty
868             \else
869                 \KV0@in@\CurrentOption\@declaredoptions
870             \ifin@
871                 \KV0@use@ption
872                 \expandafter\let\csname ds@\CurrentOption\endcsname\@empty
873             \fi
874         \fi
875     }%
876     \fi
877     \KV0@process@ptions
878 }

879 \def\KV0@in@#1#2{%
880     \in@false
881     \begingroup
882         \@for\x:=#2\do{%
883             \ifx\x#1\relax
884                 \in@true
885             \fi
886         }%
887     \edef\x{\endgroup
888         \ifin@
889             \noexpand\in@true
890         \fi
891     }%
892     \x
893 }

894 \def\KV0@process@ptions{%
895     \@for\CurrentOption:=\@curroptions\do{%
896         \@ifundefined{ds@\KV0@SanitizedCurrentOption}{%
897             \KV0@use@ption
898             \default@ds
899         }%
900         \KV0@use@ption
901     }%
902     \@for\CurrentOption:=\@declaredoptions\do{%
903         \expandafter\let\csname ds@\CurrentOption\endcsname\relax
904     }%
905     \let\CurrentOption\@empty
906     \let\@fileswith@ptions\@fileswith@ptions
907     \AtEndOfPackage{\let\@unprocessedoptions\relax}%
908 }

909 \def\KV0@use@ption{%
910     \begingroup
911         \edef\x{\endgroup
912             \noexpand\@removeelement{%

```

```

913         \detokenize\expandafter{\CurrentOption}%
914     }{%
915         \detokenize\expandafter{\@unusedoptionlist}%
916     }%
917 }%
918 \x\@unusedoptionlist
919 \csname ds@KVO@SanitizedCurrentOption\endcsname
920 }
921 \def\OptionNotUsed{%
922     \ifx\@current\@clsextension
923         \xdef\@unusedoptionlist{%
924             \ifx\@unusedoptionlist\@empty
925                 \else
926                     \detokenize\expandafter{\@unusedoptionlist,}%
927                 \fi
928                 \detokenize\expandafter{\CurrentOption}%
929             }%
930         \fi
931     }

```

Variant of \ExecuteOptions that better protects \CurrentOption.

```

932 \def\CurrentOption@SaveLevel{0}
933 \def\ExecuteOptions{%
934     \expandafter\KVO@ExecuteOptions
935     \csname CurrentOption@\CurrentOption@SaveLevel\endcsname
936 }
937 \def\KVO@ExecuteOptions#1#2{%
938     \let#1\CurrentOption
939     \edef\CurrentOption@SaveLevel{%
940         \the\numexpr\CurrentOption@SaveLevel+1%
941     }%
942     \@for\CurrentOption:=#2\do{%
943         \csname ds@\CurrentOption\endcsname
944     }%
945     \edef\CurrentOption@SaveLevel{%
946         \the\numexpr\CurrentOption@SaveLevel-1%
947     }%
948     \let\CurrentOption#1%
949 }
950 \def\KVO@fileswith@ptions#1[#2]#3[#4]{%
951     \ifx#1\@clsextension
952         \ifx\@classoptionslist\relax
953             \KVO@normalize\KVO@temp{#2}%
954             \expandafter\gdef\expandafter\@classoptionslist\expandafter{%
955                 \KVO@temp
956             }%
957             \def\reserved@a{%
958                 \KVO@onefilewithoptions{#3}[#{#2}][#{#4}]#1%
959                 \@documentclasshook
960             }%
961         \else
962             \def\reserved@a{%
963                 \KVO@onefilewithoptions{#3}[#{#2}][#{#4}]#1%
964             }%
965         \fi
966     \else
967         \begingroup
968         \let\KVO@temp\relax
969         \let\KVO@onefilewithoptions\relax
970         \let\@pkgextension\relax
971         \def\reserved@b##1,{%
972             \ifx\@nil##1\relax
973                 \else

```

```

974         \ifx\relax##1\relax
975         \else
976             \KV0@onefilewithoptions{##1}{\KV0@temp}{[#4]}%
977             \pkgextension
978         \fi
979         \expandafter\reserved@b
980     \fi
981 }%
982 \edef\reserved@a{\zap@space#3 \@empty}%
983 \edef\reserved@a{\expandafter\reserved@b\reserved@a,\@nil,}%
984 \toks@{#2}%
985 \def\KV0@temp{\the\toks@}%
986 \edef\reserved@a{\endgroup \reserved@a}%
987 \fi
988 \reserved@a
989 }

990 \def\KV0@onefilewithoptions#1[#2][#3]#4{%
991     \@pushfilename
992     \xdef\@currname{#1}%
993     \global\let\@currentx#4%
994     \expandafter\let\csname\@currname.\@currentx-h@k\endcsname\@empty
995     \let\CurrentOption\@empty
996     \@reset@ptions
997     \makeatletter
998     \def\reserved@a{%
999         \ifl@aded\@currentx{#1}{%
1000             \ifOptions\@currentx{#1}{#2}{%
1001                 }{%
1002                     \begin@group
1003                     \@ifundefined{opt@#1.\@currentx}{%
1004                         \def\x{%
1005                             }{%
1006                                 \edef\x{%
1007                                     \expandafter\expandafter\expandafter\strip@prefix
1008                                     \expandafter\meaning\csname opt@#1.\@currentx\endcsname
1009                                 }%
1010                             }%
1011                             \def\y{#2}%
1012                             \edef\y{\expandafter\strip@prefix\meaning\y}%
1013                             \@latex@error{Option clash for \cls@pkg\space #1}{%
1014                                 The package #1 has already been loaded %
1015                                 with options:\MessageBreak
1016                                 \space\space[\x]\MessageBreak
1017                                 There has now been an attempt to load it %
1018                                 with options:\MessageBreak
1019                                 \space\space[\y]\MessageBreak
1020                                 Adding the global options:\MessageBreak
1021                                 \space\space
1022                                 \x,\y\MessageBreak
1023                                 to your \noexpand\documentclass declaration may fix this.%
1024                                 \MessageBreak
1025                                 Try typing \space <return> \space to proceed.%
1026                             }%
1027                             \end@group
1028                         }%
1029                     }{%
1030                         \@pass@ptions\@currentx{#2}{#1}%
1031                         \global\expandafter
1032                         \let\csname ver@\@currname.\@currentx\endcsname\@empty
1033                         \InputIfFileExists
1034                         {\@currname.\@currentx}%
1035                         {}%

```

```

1036      {\@missingfileerror\@currname\@currentx}%
1037      \let\@unprocessedoptions\@unprocessedoptions
1038      \csname\@currname.\@currentx-h@@k\endcsname
1039      \expandafter\let\csname\@currname.\@currentx-h@@k\endcsname
1040          \undefined
1041      \@unprocessedoptions
1042  }%
1043  \@ifl@ter\@currentx{#1}{#3}{%
1044  }{%
1045      \@latex@warning@no@line{%
1046          You have requested,\on@line, %
1047          version\MessageBreak
1048          #3' of \@cls@pkg\space #1,\MessageBreak
1049          but only version\MessageBreak
1050          '\csname ver@#1.\@currentx\endcsname'\MessageBreak
1051          is available%
1052      }%
1053  }%
1054  \ifx\@currentx\@clsextension\let\LoadClass\@twoloadclasserror\fi
1055  \@popfilename
1056  \@reset@options
1057  }%
1058  \reserved@a
1059  }

1060 \def\@unknownoptionerror{%
1061     \@latex@error{%
1062         Unknown option '\KV@SanitizedCurrentOption' %
1063         for \@cls@pkg\space'\@currname'%
1064     }{%
1065         The option '\KV@SanitizedCurrentOption' was not declared in %
1066         \@cls@pkg\space'\@currname', perhaps you\MessageBreak
1067         misspelled its name. %
1068         Try typing \space <return> %
1069         \space to proceed.%
1070     }%
1071 }

1072 \def\@unprocessedoptions{%
1073     \ifx\@currentx\@pkgextension
1074         \@ifundefined{opt@\@currname.\@currentx}{%
1075             \let\@curroptions\@empty
1076         }{%
1077             \expandafter\let\expandafter\@curroptions
1078                 \csname opt@\@currname.\@currentx\endcsname
1079         }%
1080         \@for\CurrentOption:=\@curroptions\do{%
1081             \ifx\CurrentOption\@empty\else\@unknownoptionerror\fi
1082         }%
1083     \fi
1084 }

1085 \def\KV@SanitizedCurrentOption{%
1086     \expandafter\strip@prefix\meaning\CurrentOption
1087 }

    Normalize option list.
1088 \def\KV@normalize#1#2{%
1089     \let\KV@result\@empty
1090     \KV@splitcomma#2,\@nil
1091     \let#1\KV@result
1092 }
1093 \def\KV@splitcomma#1,#2\@nil{%
1094     \KV@ifempty{#1}{%
1095         \KV@checkkv#1=\@nil

```

```

1096 }%
1097 \KV0@ifempty{#2}{\KV0@splitcomma#2\@nil}%
1098 }
1099 \def\KV0@ifempty#1{%
1100 \expandafter\ifx\expandafter\\detokenize{#1}\\%
1101 \expandafter\@firstoftwo
1102 \else
1103 \expandafter\@secondoftwo
1104 \fi
1105 }
1106 \def\KV0@checkkv#1=#2\@nil{%
1107 \KV0@ifempty{#2}{%
1108 % option without value
1109 \edef\KV0@x{\zap@space#1 \@empty}%
1110 \ifx\KV0@x\@empty
1111 % ignore empty option
1112 \else
1113 % append to list
1114 \edef\KV0@result{%
1115 \etex@unexpanded\expandafter{\KV0@result},\KV0@x
1116 }%
1117 \fi
1118 }{%
1119 % #1: "key", #2: "value="
1120 % add key part
1121 \edef\KV0@result{%
1122 \etex@unexpanded\expandafter{\KV0@result},%
1123 \zap@space#1 \@empty
1124 }%
1125 \futurelet\@let@token\KV0@checkfirsttok#2 \@nil| = \@nil|\KV0@nil
1126 }%
1127 }
1128 \def\KV0@checkfirsttok{%
1129 \ifx\@let@token\bgroup
1130 % no space at start
1131 \expandafter\KV0@removelastspace\expandafter=%
1132 % "<value><spaceopt>= \@nil"
1133 \else
1134 \expandafter\KV0@checkfirstA
1135 \fi
1136 }
1137 \def\KV0@checkfirstA#1 #2\@nil{%
1138 \KV0@ifempty{#2}{%
1139 \KV0@removelastspace=#1 \@nil
1140 }{%
1141 \KV0@ifempty{#1}{%
1142 \KV0@removelastspace=#2\@nil
1143 }{%
1144 \KV0@removelastspace=#1 #2\@nil
1145 }%
1146 }%
1147 }
1148 \def\KV0@removelastspace#1 = \@nil|#2\KV0@nil{%
1149 \KV0@ifempty{#2}{%
1150 \edef\KV0@result{%
1151 \etex@unexpanded\expandafter{\KV0@result}%
1152 \etex@unexpanded\expandafter{\KV0@removegarbage#1\KV0@nil}%
1153 }%
1154 }{%
1155 \edef\KV0@result{%
1156 \etex@unexpanded\expandafter{\KV0@result}%
1157 \etex@unexpanded{#1}%

```

```

1158 }%
1159 }%
1160 }
1161 \def\KV0@removegarbage#1= \@nil#2\KV0@nil{#1}%
    Arguments #1 and #2 are macros.
1162 \def\KV0@removeelement#1#2{%
1163   \begingroup
1164   \toks@={}%
1165   \@for\x:=#2\do{%
1166     \ifx\x\@empty
1167     \else
1168       \ifx\x#1\relax
1169       \else
1170         \edef\t{\the\toks@}%
1171         \ifx\t\@empty
1172         \else
1173           \toks@\expandafter{\the\toks@,}%
1174         \fi
1175         \toks@\expandafter{\the\expandafter\toks@\x}%
1176       \fi
1177     \fi
1178   }%
1179   \edef\x{\endgroup
1180     \def\noexpand#2{\the\toks@}%
1181   }%
1182   \x
1183 }
1184 \let\@@fileswith@pti@ns\KV0@fileswith@pti@ns
1185 \ifx\@fileswith@pti@ns\@badrequireerror
1186 \else
1187   \let\@fileswith@pti@ns\KV0@fileswith@pti@ns
1188 \fi
\KV0@Patch
1189 \let\KV0@Patch=Y
1190 \KV0@AtEnd
1191 </patch>

```

## 7 Test

### 7.1 Preface for standard catcode check

```

1192 <*test1>
1193 \input miniltx.tex\relax
1194 </test1>

```

### 7.2 Catcode checks for loading

```

1195 <*test1>
1196 \catcode'\{=1 %
1197 \catcode'\}=2 %
1198 \catcode'\#=6 %
1199 \catcode'\@=11 %
1200 \expandafter\ifx\csname count@\endcsname\relax
1201   \countdef\count@=255 %
1202 \fi
1203 \expandafter\ifx\csname @gobble\endcsname\relax
1204   \long\def\@gobble#1{%
1205 \fi
1206 \expandafter\ifx\csname @firstofone\endcsname\relax
1207   \long\def\@firstofone#1{#1}%

```



```

1208 \fi
1209 \expandafter\ifx\csname loop\endcsname\relax
1210 \expandafter\@firstofone
1211 \else
1212 \expandafter\@gobble
1213 \fi
1214 {%
1215 \def\loop#1\repeat{%
1216 \def\body{#1}%
1217 \iterate
1218 }%
1219 \def\iterate{%
1220 \body
1221 \let\next\iterate
1222 \else
1223 \let\next\relax
1224 \fi
1225 \next
1226 }%
1227 \let\repeat=\fi
1228 }%
1229 \def\RestoreCatcodes{}
1230 \count@=0 %
1231 \loop
1232 \edef\RestoreCatcodes{%
1233 \RestoreCatcodes
1234 \catcode\the\count@=\the\catcode\count@\relax
1235 }%
1236 \ifnum\count@<255 %
1237 \advance\count@ 1 %
1238 \repeat
1239
1240 \def\RangeCatcodeInvalid#1#2{%
1241 \count@=#1\relax
1242 \loop
1243 \catcode\count@=15 %
1244 \ifnum\count@<#2\relax
1245 \advance\count@ 1 %
1246 \repeat
1247 }
1248 \expandafter\ifx\csname LoadCommand\endcsname\relax
1249 \def\LoadCommand{\input kvoptions.sty\relax}%
1250 \fi
1251 \def\Test{%
1252 \RangeCatcodeInvalid{0}{47}%
1253 \RangeCatcodeInvalid{58}{64}%
1254 \RangeCatcodeInvalid{91}{96}%
1255 \RangeCatcodeInvalid{123}{255}%
1256 \catcode'\@=12 %
1257 \catcode'\=0 %
1258 \catcode'\{=1 %
1259 \catcode'\}=2 %
1260 \catcode'\#=6 %
1261 \catcode'\[=12 %
1262 \catcode'\]=12 %
1263 \catcode'\%=14 %
1264 \catcode'\ =10 %
1265 \catcode13=5 %
1266 \LoadCommand
1267 \RestoreCatcodes
1268 }
1269 \Test

```

```

1270 \csname @@end\endcsname
1271 \end

1272 </test1>

1273 <*test2>
1274 \NeedsTeXFormat{LaTeX2e}
1275 \makeatletter
1276 \catcode'\@=11 %
1277 \def\RestoreCatcodes{}
1278 \count@=0 %
1279 \loop
1280   \edef\RestoreCatcodes{%
1281     \RestoreCatcodes
1282     \catcode\the\count@=\the\catcode\count@\relax
1283   }%
1284 \ifnum\count@<255 %
1285   \advance\count@\@ne
1286 \repeat
1287
1288 \def\RangeCatcodeInvalid#1#2{%
1289   \count@=#1\relax
1290   \loop
1291     \catcode\count@=15 %
1292   \ifnum\count@<#2\relax
1293     \advance\count@\@ne
1294   \repeat
1295 }
1296 \def\Test#1{%
1297   \RangeCatcodeInvalid{0}{47}%
1298   \RangeCatcodeInvalid{58}{64}%
1299   \RangeCatcodeInvalid{91}{96}%
1300   \RangeCatcodeInvalid{123}{255}%
1301   \catcode'\@=12 %
1302   \catcode'\=0 %
1303   \catcode'\{=1 %
1304   \catcode'\}=2 %
1305   \catcode'\#=6 %
1306   \catcode'\[=12 %
1307   \catcode'\]=12 %
1308   \catcode'\%=14 %
1309   \catcode'\ =10 %
1310   \catcode13=5 %
1311   #1\relax
1312   \RestoreCatcodes
1313 }
1314 \Test{\RequirePackage{kvoptions-patch}}%
1315 \Test{\RequirePackage{kvoptions}}%
1316 \csname @@end\endcsname
1317 </test2>

```

## 8 Installation

### 8.1 Download

**Package.** 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.

---

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

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

[CTAN:install/macros/latex/contrib/oberdiek.tds.zip](#)

*TDS* refers to the standard “A Directory Structure for 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 `mktexlsr`.

## 8.5 Some details for the interested

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

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

**Unpacking with  $\LaTeX$ .** The `.dtx` chooses its action depending on the format:

**plain- $\TeX$ :** Run `docstrip` and extract the files.

**$\LaTeX$ :** Generate the documentation.

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

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

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

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

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

An example follows how to generate the documentation with pdf $\LaTeX$ :

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

## 9 References

- [1] Package `ifthen`, David Carlisle, 2001/05/26.[CTAN:macros/latex/base/ifthen.dtx](#)
- [2] Package `helvet`, Sebastian Rahtz, Walter Schmidt, 2004/01/26.[CTAN:macros/latex/required/psnfss/psfonts.dtx](#)
- [3] Package `hyperref`, Sebastian Rahtz, Heiko Oberdiek, 2006/02/12.[CTAN:macros/latex/contrib/hyperref/](#)
- [4] Package `keyval`, David Carlisle, 1999/03/16.[CTAN:macros/latex/required/graphics/keyval.dtx](#)
- [5] Package `multicol`, Frank Mittelbach, 2004/02/14.[CTAN:macros/latex/required/tools/multicol.dtx](#)
- [6] Package `tabularx`, David Carlisle, 1999/01/07.[CTAN:macros/latex/required/tools/tabularx.dtx](#)
- [7] Package `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  $\LaTeX$ 3 Project,  *$\LaTeX$ 2 $\epsilon$  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	
\#	1198, 1260, 1305
\%	1263, 1308
\@	1199, 1256, 1276, 1301
\@@fileswith@pti@ns	906, 1184
\@unprocessedoptions	1037, 1072
\@SetupDriver	41, 43
\@badrequireerror	1185
\@classoptionslist	591, 593, 848, 866, 952, 954
\@cls@pkg	1013, 1048, 1063, 1066
\@clsextension	271, 272, 291, 311, 312, 336, 349, 382, 410, 411, 432, 442, 463, 543, 555, 566, 589, 631, 845, 864, 922, 951, 1054
\@current	218, 229, 232, 235, 240, 243, 246, 271, 291, 292, 311, 382, 410, 450, 589, 625, 629, 631, 658, 700, 829, 835, 845, 864, 922, 993, 994, 999, 1000, 1003, 1008, 1030, 1032, 1034, 1036, 1038, 1039, 1043, 1050, 1054, 1073, 1074, 1078
\@currname	57, 215, 229, 232, 233, 235, 240, 243, 244, 246, 270, 292, 310, 387, 409, 450, 625, 629, 658, 700, 829, 835, 992, 994, 1032, 1034, 1036, 1038, 1039, 1063, 1066, 1074, 1078
\@curroptions	830, 833, 850, 895, 1075, 1077, 1080
\@declaredoptions	841, 869, 902
\@documentclasshook	959
\@ehc	298, 478, 527, 563
\@ehd	787
\@empty	88, 89, 460, 509, 588, 641, 686, 694, 697, 707, 711, 828, 830, 842, 857, 867, 872, 905, 924, 982, 994, 995, 1032, 1075, 1081, 1089, 1109, 1110, 1123, 1166, 1171
\@expandtwoargs	617
\@fileswith@pti@ns	906, 1185, 1187
\@firstofone	331, 334, 402, 1207, 1210
\@firstoftwo	1101
\@for	593, 634, 665, 696, 841, 866, 882, 895, 902, 942, 1080, 1165
\@gobble	345, 505, 721, 1204, 1212
\@gobbletwo	400
\@if@pti@ns	802
\@if@ptions	798, 1000
\@ifdefinable	325
\@ifl@aded	999
\@ifl@ter	1043
\@ifnextchar	363
\@ifpackageloaded	790
\@ifstar	577, 721, 838
\@ifundefined	197, 200, 205, 208, 214, 217, 232, 243, 258, 285, 473, 594, 625, 635, 658, 666, 814, 829, 896, 1003, 1074
\@latex@error	1013, 1061
\@latex@warning@no@line	1045
\@let@token	1125, 1129
\@missingfileerror	1036
\@namedef	371, 421, 450
\@nameuse	700
\@ne	220, 1285, 1293
\@nil	595, 636, 667, 689, 699, 705, 712, 715, 972, 983, 1090, 1093, 1095, 1097, 1106, 1125, 1132, 1137, 1139, 1142, 1144, 1148, 1161
\@onelevel@sanitize	201, 329, 360, 361, 608, 639
\@pass@ptions	812, 1030
\@pkgextension	274, 314, 413, 467, 970, 977, 1073
\@popfilename	1055
\@pushfilename	991
\@removeelement	617, 912
\@reset@ptions	996, 1056
\@secondoftwo	1103
\@tempc	587
\@twoloadclasserror	1054
\@typeset@protect	212
\@undefined	1040
\@unknownoptionerror	68, 1060, 1081
\@unprocessedoptions	687, 907, 1037, 1041
\@unusedoptionlist	618, 641, 642, 645, 647, 915, 918, 923, 924, 926
\@x@protect	209
\[	1261, 1306
\	373, 1100, 1257, 1302
\{	1196, 1258, 1303

<code>\}</code> .....	1197, 1259, 1304	<code>\DeclareDefaultOption</code> ....	6, 55, 449
<code>\]</code> .....	1262, 1307	<code>\DeclareOption</code> .....	220, 221
		<code>\DeclareStringOption</code> .....	4, 26, 28, 50, 52, 362, 458
<code>\_</code> .....	1264, 1309	<code>\DeclareVoidOption</code> ..	5, 32, 33, 34, 398
<b>A</b>		<code>\default@ds</code> .....	898
<code>\advance</code> .....	1237, 1245, 1285, 1293	<code>\define@key</code> .....	227, 238, 269, 309, 379, 408, 461, 465, 530
<code>\AtEndOfPackage</code> .....	222, 687, 907	<code>\detokenize</code> .....	803, 807, 913, 915, 926, 928, 1100
<b>B</b>		<code>\DisableKeyvalOption</code> .....	6, 469
<code>\body</code> .....	1216, 1220	<code>\do</code> .....	593, 634, 665, 696, 841, 866, 882, 895, 902, 942, 1080, 1165
<b>C</b>		<code>\documentclass</code> .....	786, 1023
<code>\catcode</code> ..	100, 101, 102, 103, 104, 115, 116, 117, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 159, 160, 163, 164, 165, 166, 170, 171, 172, 173, 177, 179, 220, 729, 730, 733, 734, 735, 736, 740, 741, 742, 743, 747, 749, 1196, 1197, 1198, 1199, 1234, 1243, 1256, 1257, 1258, 1259, 1260, 1261, 1262, 1263, 1264, 1265, 1276, 1282, 1291, 1301, 1302, 1303, 1304, 1305, 1306, 1307, 1308, 1309, 1310	<code>\ds@</code> .....	828
<code>\ClassError</code> .....	556	<b>E</b>	
<code>\ClassInfo</code> .....	350, 383, 443, 544	<code>\emph</code> .....	48, 87, 91, 93
<code>\ClassWarning</code> .....	337, 433, 567	<code>\empty</code> .....	109
<code>\count@</code> .....	1201, 1230, 1234, 1236, 1237, 1241, 1243, 1244, 1245, 1278, 1282, 1284, 1285, 1289, 1291, 1292, 1293	<code>\end</code> .....	1271
<code>\countdef</code> .....	1201	<code>\endcsname</code> .....	105, 118, 142, 155, 162, 229, 235, 240, 246, 256, 257, 265, 303, 304, 305, 306, 325, 355, 392, 417, 480, 488, 492, 504, 508, 531, 536, 537, 539, 540, 629, 719, 732, 774, 804, 816, 821, 823, 835, 857, 872, 903, 919, 935, 943, 994, 1008, 1032, 1038, 1039, 1050, 1078, 1200, 1203, 1206, 1209, 1248, 1270, 1316
<code>\csname</code> ..	105, 118, 142, 155, 162, 228, 235, 239, 246, 256, 257, 265, 303, 304, 305, 306, 325, 355, 392, 417, 480, 488, 492, 504, 508, 531, 536, 537, 539, 540, 629, 719, 732, 774, 804, 815, 820, 823, 835, 857, 872, 903, 919, 935, 943, 994, 1008, 1032, 1038, 1039, 1050, 1078, 1200, 1203, 1206, 1209, 1248, 1270, 1316	<code>\endinput</code> .....	127, 769
<code>\CurrentOption</code> ....	35, 37, 38, 41, 58, 62, 64, 426, 632, 634, 636, 639, 642, 648, 653, 663, 665, 667, 670, 674, 686, 696, 697, 699, 708, 712, 841, 842, 852, 857, 866, 867, 869, 872, 895, 902, 903, 905, 913, 928, 938, 942, 943, 948, 995, 1080, 1081, 1086	<code>\etex@unexpanded</code> .....	599, 603, 1115, 1122, 1151, 1152, 1156, 1157
<code>\CurrentOption@SaveLevel</code> .....	932, 935, 939, 940, 945, 946	<code>\ExecuteOptions</code> .....	933
<code>\CurrentOptionKey</code> .....	708, 711	<b>F</b>	
<code>\CurrentOptionValue</code> .....	56, 706, 707, 709, 716	<code>\futurelet</code> .....	1125
<b>D</b>		<b>G</b>	
<code>\DeclareBoolOption</code> .	4, 20, 22, 252, 456	<code>\gdef</code> .....	644, 815, 820, 954
<code>\DeclareComplementaryOption</code> ....	5, 284, 457	<b>I</b>	
		<code>\ifcase</code> .....	106
		<code>\ifetex@unexpanded</code> .....	781
		<code>\ifin@</code> .....	855, 870, 888
		<code>\ifKV@dyn@global</code> .....	486, 490, 497, 502, 506, 513, 534, 550
		<code>\ifMCS@print</code> .....	79
		<code>\ifnum</code> .....	1236, 1244, 1284, 1292
		<code>\ifpdf</code> .....	25
		<code>\ifx</code> .....	56, 88, 107, 109, 118, 142, 150, 271, 291, 311, 330, 333, 336, 349, 373, 382, 410, 430, 432, 442, 523, 543, 555, 566, 589, 591, 597, 631, 638, 641, 694, 697, 707, 719, 774, 842, 845, 864, 867, 883, 922, 924, 951, 952, 972, 974, 1054, 1073, 1081, 1100, 1110, 1129, 1166, 1168, 1171, 1185, 1200, 1203, 1206, 1209, 1248
		<code>\immediate</code> .....	120, 144

\in@	852	\KVO@splitcomma	1090, 1093, 1097
\in@false	880	\KVO@temp	588, 598, 600, 610, 611, 626, 633, 660, 664, 681, 685, 800, 807, 813, 817, 823, 953, 955, 968, 976, 985
\in@true	884, 889	\KVO@true	330, 358
\input	1193, 1249	\KVO@use@option	856, 871, 897, 900, 909
\InputIfFileExists	1033	\KVO@VOID@	408, 424, 430
\iterate	1217, 1219, 1221	\KVO@voidkey	409, 425
<b>K</b>		\KVO@x	1109, 1110, 1115
\KV@sp@def	716	\KVO@xprocess@options	838, 863
\KV@setcurrentvalue	715	\KV@dyn@action	473, 476, 480
\KVO@action@error	516	\KV@dyn@ext	460, 463, 467, 532, 543
\KVO@action@ignore	500	\KV@dyn@name	459, 462, 466, 523, 532, 548
\KVO@action@undef	484	<b>L</b>	
\KVO@action@warning	519	\LoadClass	1054
\KVO@AtEnd	175, 176, 724, 745, 746, 767, 768, 778, 788, 796, 1190	\LoadCommand	1249, 1266
\KVO@boolkey	270, 310, 327	\loop	1215, 1231, 1242, 1279, 1290
\KVO@calldefault	682, 690	<b>M</b>	
\KVO@checkfirstA	1134, 1137	\makeatletter	997, 1275
\KVO@checkfirsttok	1125, 1128	\MCS@driver	84
\KVO@checkkv	1095, 1106	\MCS@emph	88, 94
\KVO@CurrentOption	593, 595, 604, 608, 613, 617	\meaning	202, 476, 1008, 1012, 1086
\KVO@DeclareStringOption	364, 366, 369	\MessageBreak	58, 260, 261, 287, 292, 294, 296, 297, 342, 438, 476, 496, 512, 525, 549, 561, 572, 784, 792, 793, 1015, 1016, 1018, 1019, 1020, 1022, 1024, 1047, 1048, 1049, 1050, 1066
\KVO@disable@error	554	<b>N</b>	
\KVO@disable@warning	565	\NeedsTeXFormat	4, 727, 1274
\KVO@do@action	517, 520, 522	\newcommand	40, 43, 249, 252, 284, 362, 398, 449, 469, 576
\KVO@ExecuteOptions	934, 937	\next	400, 402, 405, 1221, 1223, 1225
\KVO@false	333, 358	\numexpr	940, 946
\KVO@family	227, 269, 293, 309, 379, 408, 580	<b>O</b>	
\KVO@fileswith@ptions	950, 1184, 1187	\on@line	1046
\KVO@getkey	594, 636, 667, 689	\OptionNotUsed	921
\KVO@ifdefinable	253, 254, 255, 301, 302, 324, 370, 401	<b>P</b>	
\KVO@ifempty	1094, 1097, 1099, 1107, 1138, 1141, 1149	\PackageError	286, 474, 524, 558, 783
\KVO@in@	869, 879	\PackageInfo	123, 352, 385, 445, 495, 511, 546
\KVO@nil	1125, 1148, 1152, 1161	\PackageWarning	259, 339, 435, 569
\KVO@normalize	800, 813, 953, 1088	\PackageWarningNoLine	57, 775, 791
\KVO@onefilewithoptions	958, 963, 969, 976, 990	\PassOptionsToPackage	63, 80
\KVO@param	328, 329, 330, 333, 342, 354, 355	\ProcessKeyvalOptions	3, 74, 576, 720
\KVO@Patch	597, 638, 1189	\ProcessOptions	226, 827
\KVO@prefix	238, 253, 254, 255, 257, 258, 265, 276, 285, 295, 301, 302, 303, 304, 305, 306, 316, 370, 371, 392, 401, 417, 421	\protect	210
\KVO@process@ptions	861, 877, 894	\ProvidesPackage	5, 156, 771
\KVO@process@options	838, 840	<b>R</b>	
\KVO@ProcessKeyvalOptions	580, 584, 586	\RangeCatcodeInvalid	1240, 1252, 1253, 1254, 1255, 1288, 1297, 1298, 1299, 1300
\KVO@removeelement	1162	\renewcommand	93
\KVO@removegarbage	1152, 1161	\repeat	1215, 1227, 1238, 1246, 1286, 1294
\KVO@removelastspace	1131, 1139, 1142, 1144, 1148	\RequirePackage	7, 8, 84, 198, 223, 780, 785, 1314, 1315
\KVO@result	1089, 1091, 1114, 1115, 1121, 1122, 1150, 1151, 1155, 1156	\reserved@a	957, 962, 982, 983, 986, 988, 998, 1058
\KVO@SanitizedCurrentOption	896, 919, 1062, 1065, 1085		
\KVO@setcurrents	699, 705		
\KVO@setcurrentvalue	712, 715		



<code>\reserved@b</code> . . . . .	971, 979, 983	<code>\toks@</code> . . . . .	374, 376, 379, 381,
<code>\RestoreCatcodes</code> . . . .	1229, 1232,		388, 626, 628, 632, 633, 652,
	1233, 1267, 1277, 1280, 1281, 1312		653, 659, 660, 663, 664, 673,
			674, 683, 846, 848, 852, 984,
			985, 1164, 1170, 1173, 1175, 1180
<b>S</b>			
<code>\setkeys</code> . . . . .	44, 250, 471, 683	<code>\tw@</code> . . . . .	624, 669, 670, 682, 850, 852
<code>\SetupDriver</code> . . . .	32, 33, 34, 35, 38, 40		
<code>\SetupKeyvalOptions</code> . .	3, 13, 249, 452	<b>U</b>	
<code>\space</code> .	291, 1013, 1016, 1019, 1021,	<code>\unexpanded</code> . . . . .	784
	1025, 1048, 1063, 1066, 1068, 1069		
<code>\strip@prefix</code> . . . . .		<b>W</b>	
	.. 202, 206, 476, 1007, 1012, 1086	<code>\write</code> . . . . .	120, 144
<b>T</b>			
<code>\t</code> . . . . .	1170, 1171	<b>X</b>	
<code>\Test</code> . . . .	1251, 1269, 1296, 1314, 1315	<code>\x</code> . . . . .	105, 107, 109, 119, 123,
<code>\textcolor</code> . . . . .	94		125, 143, 148, 155, 161, 169,
<code>\the</code> . . .	163, 164, 165, 166, 177, 379,		268, 279, 308, 319, 378, 395,
	388, 632, 653, 660, 663, 670,		407, 420, 428, 430, 472, 482,
	674, 682, 683, 733, 734, 735,		485, 501, 529, 579, 582, 692,
	736, 747, 852, 940, 946, 985,		694, 731, 739, 801, 810, 851,
	1170, 1173, 1175, 1180, 1234, 1282		854, 882, 883, 887, 892, 911,
<code>\TMP@EnsureCode</code> . . .	174, 181, 182,		918, 1004, 1006, 1016, 1022,
	183, 184, 185, 186, 187, 188,		1165, 1166, 1168, 1175, 1179, 1182
	189, 190, 191, 192, 193, 194,	<b>Y</b>	
	195, 196, 744, 751, 752, 753,	<code>\y</code> . . . . .	1011, 1012, 1019, 1022
	754, 755, 756, 757, 758, 759,		
	760, 761, 762, 763, 764, 765, 766	<b>Z</b>	
<code>\toks</code> . . . . .	624, 669, 670, 682, 850, 852	<code>\zap@space</code> . . . . .	711, 982, 1109, 1123