

# The kvoptions package

Heiko Oberdiek\*

<heiko.oberdiek at gmail.com>

2016/05/16 v3.12

## 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>3</b>
1.1	The beginning . . . . .	3
1.2	Overview . . . . .	3
<b>2</b>	<b>Usage</b>	<b>4</b>
2.1	Process options . . . . .	4
2.1.1	<code>\ProcessKeyvalOptions</code> . . . . .	4
2.1.2	<code>\ProcessLocalKeyvalOptions</code> . . . . .	4
2.1.3	<code>\SetupKeyvalOptions</code> . . . . .	5
2.2	Option declarations . . . . .	5
2.2.1	<code>\DeclareStringOption</code> . . . . .	5
2.2.2	<code>\DeclareBoolOption</code> . . . . .	6
2.2.3	<code>\DeclareComplementaryOption</code> . . . . .	7
2.2.4	<code>\DeclareVoidOption</code> . . . . .	7
2.2.5	<code>\DeclareDefaultOption</code> . . . . .	7
2.2.6	Local options . . . . .	8
2.2.7	Dynamic options . . . . .	8
2.2.8	<code>\DisableKeyvalOption</code> . . . . .	8
2.2.9	<code>\AddToKeyvalOption</code> . . . . .	9
2.3	Global vs. local options . . . . .	9
2.4	Summary of internal macros . . . . .	10
2.5	plain <code>T<sub>E</sub>X</code> . . . . .	10
<b>3</b>	<b>Example</b>	<b>10</b>
<b>4</b>	<b>Package options</b>	<b>12</b>
4.1	Package <code>kvoptions-patch</code> . . . . .	12
4.2	Option <code>debugshow</code> . . . . .	13

---

\*Please report any issues at <https://github.com/ho-tex/oberdiek/issues>

<b>5</b>	<b>Limitations</b>	<b>14</b>
5.1	Compatibility	14
5.1.1	Package <code>kvoptions-patch</code> vs. <code>package xkvltxp</code>	14
5.2	Limitations	14
5.2.1	Option comparisons	14
5.2.2	Option list parsing with <code>package kvoptions-patch</code>	15
<b>6</b>	<b>Implementation</b>	<b>15</b>
6.1	Preamble	15
6.2	Option declaration macros	18
6.2.1	<code>\SetupKeyvalOptions</code>	18
6.2.2	<code>\DeclareBoolOption</code>	19
6.2.3	<code>\DeclareStringOption</code>	21
6.2.4	<code>\DeclareVoidOption</code>	22
6.2.5	<code>\DeclareDefaultOption</code>	23
6.2.6	<code>\DeclareLocalOptions</code>	23
6.3	Dynamic options	24
6.3.1	<code>\DisableKeyvalOption</code>	24
6.4	Change option code	26
6.4.1	<code>\AddToKeyvalOption</code>	26
6.5	Process options	27
6.5.1	Get global options	27
6.5.2	<code>\ProcessKeyvalOptions</code>	27
6.5.3	<code>\ProcessLocalKeyvalOptions</code>	30
6.5.4	Helper macros	31
6.6	plain <code>T<sub>E</sub>X</code>	32
6.7	Package <code>kvoptions-patch</code>	32
<b>7</b>	<b>Test</b>	<b>40</b>
7.1	Preface for standard catcode check	40
7.2	Catcode checks for loading	41
<b>8</b>	<b>Installation</b>	<b>44</b>
8.1	Download	44
8.2	Bundle installation	45
8.3	Package installation	45
8.4	Refresh file name databases	45
8.5	Some details for the interested	46
<b>9</b>	<b>Catalogue</b>	<b>46</b>
<b>10</b>	<b>References</b>	<b>47</b>
<b>11</b>	<b>History</b>	<b>47</b>
	[0000/00/00 v0.0]	47
	[2004/02/22 v1.0]	47
	[2006/02/16 v2.0]	48
	[2006/02/20 v2.1]	48
	[2006/06/01 v2.2]	48
	[2006/08/17 v2.3]	48
	[2006/08/22 v2.4]	48
	[2007/04/11 v2.5]	48
	[2007/05/06 v2.6]	48
	[2007/06/11 v2.7]	48

[2007/10/02 v2.8]	48
[2007/10/11 v2.9]	49
[2007/10/18 v3.0]	49
[2009/04/10 v3.1]	49
[2009/07/17 v3.2]	49
[2009/07/21 v3.3]	49
[2009/08/13 v3.4]	49
[2009/12/04 v3.5]	49
[2009/12/08 v3.6]	49
[2010/02/22 v3.7]	49
[2010/07/23 v3.8]	49
[2010/12/02 v3.9]	49
[2010/12/23 v3.10]	49
[2011/06/30 v3.11]	50
[2016/05/16 v3.12]	50
<b>12 Index</b>	<b>50</b>

# 1 Introduction

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

## 1.1 The beginning

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

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

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

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

## 1.2 Overview

Package `kvoptions` connects package `keyval` with L<sup>A</sup>T<sub>E</sub>X’s package and class `options`:

Package <b>keyval</b>	Package <b>kvoptions</b>	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`

<code>\ProcessKeyvalOptions {<i>&lt;family&gt;</i>}</code> <code>\ProcessKeyvalOptions *</code>
--

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.

#### 2.1.2 `\ProcessLocalKeyvalOptions`

<code>\ProcessLocalKeyvalOptions {<i>&lt;family&gt;</i>}</code> <code>\ProcessLocalKeyvalOptions *</code>
--

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

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

### 2.1.3 \SetupKeyvalOptions

```
\SetupKeyvalOptions {
  family = <family>,
  prefix = <prefix>
  setkeys = <setkeys command>
}
```

This command allows to configure the default assumptions that are based on the current package or class name. L<sup>A</sup>T<sub>E</sub>X remembers this name in `\@currname`. The syntax description of the default looks a little weird, therefor an example is given for a package or class named `foobar`.

Key	Default	(example)	Used by
family	<code>\@currname</code>	( <code>foobar</code> )	<code>\ProcessKeyvalOptions*</code> <code>\DeclareBoolOption</code> <code>\DeclareStringOption</code>
prefix	<code>\@currname</code> @	( <code>foobar@</code> )	<code>\DeclareBoolOption</code> <code>\DeclareStringOption</code> <code>\DeclareVoidOption</code>
setkeys	<code>\setkeys</code>	( <code>\kvsetkeys</code> )	<code>\ProcessKeyvalOptions</code> <code>\ProcessLocalKeyvalOptions</code>

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

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

## 2.2 Option declarations

The options for `\ProcessKeyvalOptions` are defined by `keyval`'s `\define@key`. Common purposes of such keys are boolean switches, they enable or disable something. Or they store a name or some kind of string in a macro. The following commands help the user. He declares what he wants and `kvoptions` take care of the key definition, resource allocation and initialization.

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

### 2.2.1 \DeclareStringOption

```
\DeclareStringOption [init] {<key>} [default]
```

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

The the option  $\langle key \rangle$  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  $\langle key \rangle$  is defined with the default  $\langle default \rangle$ .

Example for a package with the following two lines:

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

Then `\DeclareStringOption` defines the macro with content `me`, note L<sup>A</sup>T<sub>E</sub>X complains if the name of the macro already exists:

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

The option definition is similar to:

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

### 2.2.2 \DeclareBoolOption

`\DeclareBoolOption [ $\langle init \rangle$ ] { $\langle key \rangle$ }`

A boolean switch is generated, initialized by value  $\langle init \rangle$  and the corresponding key  $\langle key \rangle$  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.:

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

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

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

### 2.2.3 `\DeclareComplementaryOption`

`\DeclareComplementaryOption{<key>}{<parent>}`

Sometimes contrasting names are used to characterize the two states of a boolean switch, for example `draft` vs. `final`. Both options behave like boolean options but they do not need two 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 2.1.3). If the option is set, the macro will be executed. During the execution `\CurrentOption` is available with the current key name.

### 2.2.5 `\DeclareDefaultOption`

`\DeclareDefaultOption{<code>}`

This command does not define a specific key, it is the equivalent to 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 Local options

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

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

### 2.2.7 Dynamic options

Options of 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.

### 2.2.8 `\DisableKeyvalOption`

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

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

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

**error or warning:** Use of the option will cause an error or warning message. Also these actions require that 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 `\DisableKeyvalOption` can be executed everywhere, also outside the package. Therefore the family name and the package/class name is usually unknown for `\DisableKeyvalOption`. Therefore the argument for the family name is mandatory and for some actions the package/class name must be provided.

Usually a macro would configure the option death, Example:



```

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

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

\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.

```

### 2.2.9 \AddToKeyvalOption

```

\AddToKeyvalOption{<family>}{<key>}{<code>}
\AddToKeyvalOption*{<key>}{<code>}

```

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

## 2.3 Global vs. local options

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

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

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

- `\ProcessLocalKeyvalOptions` automatically uses all options as local options. It ignores all global options.
- `\DeclareLocalOption` or `\DeclareLocalOptions` marks options as local options. `\ProcessKeyvalOptions` will then ignore global occurrences for

these local options.

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

## 2.4 Summary of internal macros

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

Declare <code>\&lt;key&gt;</code>	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

## 2.5 plain T<sub>E</sub>X

Package `keyval` is also usable in plain T<sub>E</sub>X with the help of file `miniltx.tex`. Some features of this package `kvoptions` might also be useful for plain T<sub>E</sub>X. If L<sup>A</sup>T<sub>E</sub>X 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}[2016/05/16 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 L<sup>A</sup>T<sub>E</sub>X 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`

L<sup>A</sup>T<sub>E</sub>X'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. L<sup>A</sup>T<sub>E</sub>X 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, L<sup>A</sup>T<sub>E</sub>X 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: `Mu127u1ler`. 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 package `kvoptions-patch`. It patches L<sup>A</sup>T<sub>E</sub>X's option system and tries to teach it also to handle options that are given as pairs of key and value and to prevent expansion. It can already be used at the very beginning, before `\documentclass`:

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

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

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

Some remarks:

- The patch requires  $\varepsilon$ -T<sub>E</sub>X, its `\unexpanded` feature is much too 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  $\varepsilon$ -T<sub>E</sub>X.
- The code for the patch is quite long, there are many test cases. Thus the probability for bugs is probably not too small.
- Since 2008/10/18 v3.0 package `kvoptions-patch` is available. Before option `patch` of package `kvoptions` must be used instead. I think, the solution as standalone package `kvoptions-patch` is cleaner and avoids option clashes.

## 4.2 Option `debugshow`

The name of this option follows the convention of packages `multicol`, `tabularx`, and `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  $\text{\LaTeX}$ .

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

```

124     \fi
125     \x{kvoptions}{The package is already loaded}%
126     \aftergroup\endinput
127     \fi
128   \fi
129 \endgroup%

Package identification:

130 \begingroup\catcode61\catcode48\catcode32=10\relax%
131   \catcode13=5 % ^^M
132   \endlinechar=13 %
133   \catcode35=6 % #
134   \catcode39=12 % '
135   \catcode40=12 % (
136   \catcode41=12 % )
137   \catcode44=12 % ,
138   \catcode45=12 % -
139   \catcode46=12 % .
140   \catcode47=12 % /
141   \catcode58=12 % :
142   \catcode64=11 % @
143   \catcode91=12 % [
144   \catcode93=12 % ]
145   \catcode123=1 % {
146   \catcode125=2 % }
147   \expandafter\ifx\csname ProvidesPackage\endcsname\relax
148     \def\x#1#2#3[#4]{\endgroup
149       \immediate\write-1{Package: #3 #4}%
150       \xdef#1{#4}%
151     }%
152   \else
153     \def\x#1#2[#3]{\endgroup
154       #2[{#3}]%
155       \ifx#1\@undefined
156         \xdef#1{#3}%
157       \fi
158       \ifx#1\relax
159         \xdef#1{#3}%
160       \fi
161     }%
162   \fi
163 \expandafter\x\csname ver@kvoptions.sty\endcsname
164 \ProvidesPackage{kvoptions}%
165 [2016/05/16 v3.12 Key value format for package options (H0)]%

```

### Catcodes

```

166 \begingroup\catcode61\catcode48\catcode32=10\relax%
167   \catcode13=5 % ^^M
168   \endlinechar=13 %
169   \catcode123=1 % {
170   \catcode125=2 % }
171   \catcode64=11 % @
172   \def\x{\endgroup
173     \expandafter\edef\csname KVO@AtEnd\endcsname{%
174       \endlinechar=\the\endlinechar\relax
175       \catcode13=\the\catcode13\relax
176       \catcode32=\the\catcode32\relax
177       \catcode35=\the\catcode35\relax

```



```

178 \catcode61=\the\catcode61\relax
179 \catcode64=\the\catcode64\relax
180 \catcode123=\the\catcode123\relax
181 \catcode125=\the\catcode125\relax
182 }%
183 }%
184 \x\catcode61\catcode48\catcode32=10\relax%
185 \catcode13=5 % ^^M
186 \endlinechar=13 %
187 \catcode35=6 % #
188 \catcode64=11 % @
189 \catcode123=1 % {
190 \catcode125=2 % }
191 \def\TMP@EnsureCode#1#2{%
192 \edef\KVO@AtEnd{%
193 \KVO@AtEnd
194 \catcode#1=\the\catcode#1\relax
195 }%
196 \catcode#1=#2\relax
197 }
198 \TMP@EnsureCode{1}{14}% ^^A (comment)
199 \TMP@EnsureCode{2}{14}% ^^A (comment)
200 \TMP@EnsureCode{33}{12}% !
201 \TMP@EnsureCode{39}{12}% '
202 \TMP@EnsureCode{40}{12}% (
203 \TMP@EnsureCode{41}{12}% )
204 \TMP@EnsureCode{42}{12}% *
205 \TMP@EnsureCode{44}{12}% ,
206 \TMP@EnsureCode{45}{12}% -
207 \TMP@EnsureCode{46}{12}% .
208 \TMP@EnsureCode{47}{12}% /
209 \TMP@EnsureCode{58}{12}% :
210 \TMP@EnsureCode{62}{12}% >
211 \TMP@EnsureCode{91}{12}% [
212 \TMP@EnsureCode{93}{12}% ]
213 \TMP@EnsureCode{94}{7}% ^ (superscript)
214 \TMP@EnsureCode{96}{12}% '
215 \edef\KVO@AtEnd{\KVO@AtEnd\noexpand\endinput}

```

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

```

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

```

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

```

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

```

**Provide macros for plain TeX.**

```

221 \@ifundefined{@x@protect}{%
222 \def\x@protect#1\fi#2#3{%
223 \fi\protect#1%
224 }%

```

```

225 \let\@typeset@protect\relax
226 }{}
227 \@ifundefined{@currname}{%
228 \def\@currname{}%
229 }{}
230 \@ifundefined{@currentx}{%
231 \def\@currentx{}%
232 }{}

```

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

```

233 \DeclareOption{debugshow}{\catcode\@ne=9 }
234 \DeclareOption{patch}{%
235 \AtEndOfPackage{%
236 \RequirePackage{kvoptions-patch}[2016/05/16]%
237 }%
238 }

```

Optionen auswerten:

```

239 \ProcessOptions\relax

```

## 6.2 Option declaration macros

### 6.2.1 \SetupKeyvalOptions

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

`\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.

```

240 \define@key{KV0}{family}{%
241 \expandafter\edef\csname KV0@family@%
242 \@currname.\@currentx\endcsname{#1}%
243 }
244 \def\KV0@family{%
245 \@ifundefined{KV0@family@\@currname.\@currentx}{%
246 \@currname
247 }{%
248 \csname KV0@family@\@currname.\@currentx\endcsname
249 }%
250 }

```

`\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.

```

251 \define@key{KV0}{prefix}{%
252 \expandafter\edef\csname KV0@prefix@%
253 \@currname.\@currentx\endcsname{#1}%
254 }
255 \def\KV0@prefix{%
256 \ltx@ifundefined{KV0@prefix@\@currname.\@currentx}{%

```

```

257 \currname @%
258 }{%
259 \csname KVO@prefix@\currname.\@current\endcsname
260 }%
261 }

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

```

`\KVO@setkeys`

```

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

```

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

```

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

```

### 6.2.2 `\DeclareBoolOption`

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

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

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.

```

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

```

```

297         \else
298         \noexpand\@pkgextension
299         \fi
300         {\KV0@prefix}{#2}{####1}%
301     }%
302 }%
303 \x
304 }%
305 }%
306 }%
307 }

```

`\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.

```

308 \newcommand*{\DeclareComplementaryOption}[2]{%
309   \ifundefined{ifKV0@prefix#2}{%
310     \PackageError{kvoptions}{%
311       Cannot generate option code for ‘#1’,\MessageBreak
312       parent switch ‘#2’ does not exist%
313     }{%
314       You are inside %
315       \ifx\@currentx\@clsextension class\else package\fi\space
316       ‘\@currname.\@currentx’.\MessageBreak
317       ‘\KV0@family’ is used as family %
318       for the keyval options.\MessageBreak
319       ‘\KV0@prefix’ serves as prefix %
320       for internal switch macros.\MessageBreak
321       \MessageBreak
322       \@ehc
323     }%
324   }{%
325     \KV0@ifdefinable{\KV0@prefix#1true}{%
326       \KV0@ifdefinable{\KV0@prefix#1false}{%
327         \expandafter\let\csname\KV0@prefix#1false\expandafter\endcsname
328         \csname\KV0@prefix#2true\endcsname
329         \expandafter\let\csname\KV0@prefix#1true\expandafter\endcsname
330         \csname\KV0@prefix#2false\endcsname

```

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

```

331   \begingroup
332   \edef\x{\endgroup
333     \noexpand\define@key{\KV0@family}{#1}[true]{%
334       \noexpand\KV0@boolkey{\@currname}%
335       \ifx\@currentx\@clsextension
336         \noexpand\@clsextension
337       \else
338         \noexpand\@pkgextension
339       \fi
340       {\KV0@prefix}{#1}{####1}%
341     }%
342   }%
343   \x
344 }%
345 }%
346 }%
347 }

```

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

```
348 \def\KV0@ifdefinable#1{%
349   \expandafter\@ifdefinable\csname #1\endcsname
350 }
```

`\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

351 \def\KV0@boolkey#1#2#3#4#5{%
352   \edef\KV0@param{#5}%
353   \ltx@onelevel@sanitize\KV0@param
354   \ifx\KV0@param\KV0@true
355     \expandafter\@firstofone
356   \else
357     \ifx\KV0@param\KV0@false
358       \expandafter\expandafter\expandafter\@firstofone
359     \else
360       \ifx#2\@clsextension
361         \expandafter\ClassWarning
362       \else
363         \expandafter\PackageWarning
364       \fi
365       {#1}{%
366         Value '\KV0@param' is not supported by\MessageBreak
367         option '#4'%
368       }%
369       \expandafter\expandafter\expandafter\@gobble
370     \fi
371   \fi
372   {%
373     ^^A\ifx#2\@clsextension
374     ^^A \expandafter\ClassInfo
375     ^^A\else
376     ^^A \expandafter\PackageInfo
377     ^^A\fi
378     ^^A{#1}{[option] #4=\KV0@param}%
379     \csname#3#4\KV0@param\endcsname
380   }%
381 }
```

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

```
382 \def\KV0@true{true}
383 \def\KV0@false{false}
384 \ltx@onelevel@sanitize\KV0@true
385 \ltx@onelevel@sanitize\KV0@false
```

### 6.2.3 `\DeclareStringOption`

`\DeclareStringOption`

```
386 \newcommand*{\DeclareStringOption}[2][ ]{%
387   \@ifnextchar[{%
```

```

388 \KVO@DeclareStringOption{#1}{#2}%
389 }{%
390 \KVO@DeclareStringOption{#1}{#2}{ }[]%
391 }%
392 }

```

\KVO@DeclareStringOption

```

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

```

#### 6.2.4 \DeclareVoidOption

\DeclareVoidOption

```

422 \newcommand*\DeclareVoidOption}[2]{%
423 \begingroup
424 \let\next\@gobbletwo
425 \KVO@ifdefinable{\KVO@prefix#1}{%
426 \let\next\@firstofone
427 }%
428 \expandafter\endgroup
429 \next{%
430 \begingroup
431 \edef\x{\endgroup
432 \noexpand\define@key{\KVO@family}{#1}[\KVO@VOID@]{%
433 \noexpand\KVO@voidkey{\@currname}%
434 \ifx\@current\@clsextension
435 \noexpand\@clsextension
436 \else
437 \noexpand\@pkgextension
438 \fi
439 {#1}%

```

```

440         {####1}%
441         \expandafter\noexpand\csname\KV0@prefix#1\endcsname
442     }%
443 }%
444 \x
445 \begingroup
446     \toks@{#2}%
447 \expandafter\endgroup
448 \expandafter\def
449     \csname\KV0@prefix#1\expandafter\endcsname
450     \expandafter{\the\toks@}%
451 }%
452 }
453 \def\KV0@VOID@{@VOID@}

#1 package/class name
#2 \@pkgextension/\@clsextension
\KV0@voidkey #3 key name
#4 default (@VOID@)
#5 macro with option code
454 \def\KV0@voidkey#1#2#3#4{%
455     \def\CurrentOption{#3}%
456     \begingroup
457         \def\x{#4}%
458     \expandafter\endgroup
459     \ifx\x\KV0@VOID@
460     \else
461         \ifx#2\@clsextension
462             \expandafter\ClassWarning
463         \else
464             \expandafter\PackageWarning
465         \fi
466         {#1}{%
467             Unexpected value for option ‘#3’\MessageBreak
468             is ignored%
469         }%
470     \fi
471     ^^A\ifx#2\@clsextension
472     ^^A \expandafter\ClassInfo
473     ^^A\else
474     ^^A \expandafter\PackageInfo
475     ^^A\fi
476     ^^A{#1}{[option] #3}%
477 }

```

### 6.2.5 \DeclareDefaultOption

\DeclareDefaultOption

```

478 \newcommand*{\DeclareDefaultOption}{%
479     \@namedef{KV0@default@\@currname.\@current}{%
480 }

```

### 6.2.6 \DeclareLocalOptions

\DeclareLocalOptions

```

481 \newcommand*{\DeclareLocalOptions}[1]{%
482     \comma@parse{#1}\KV0@DeclareLocalOption
483 }

```

\KV0@DeclareLocalOption

```
484 \def\KV0@DeclareLocalOption#1{%
485   \expandafter\def\csname KV0@local@KV0@family @#1\endcsname{%}%
486 }
```

## 6.3 Dynamic options

### 6.3.1 \DisableKeyvalOption

```
487 \SetupKeyvalOptions{%
488   family=KV0dyn,%
489   prefix=KV0dyn%
490 }
491 \DeclareBoolOption[true]{global}
492 \DeclareComplementaryOption{local}{global}
493 \DeclareStringOption[undef]{action}
494 \let\KV0dyn@name\relax
495 \let\KV0dyn@ext\@empty
496 \define@key{KV0dyn}{class}{%
497   \def\KV0dyn@name{#1}%
498   \let\KV0dyn@ext\@clsextension
499 }
500 \define@key{KV0dyn}{package}{%
501   \def\KV0dyn@name{#1}%
502   \let\KV0dyn@ext\@pkgextension
503 }
504 \newcommand*\DisableKeyvalOption[3][{}]{%
505   \begingroup
506     \kvsetkeys{KV0dyn}{#1}%
507     \def\x{\endgroup}%
508     \ifundefined{KV0@action@KV0dyn@action}{%
509       \PackageError{kvoptions}{%
510         Unknown disable action %
511         '\expandafter\strip@prefix\meaning\KV0dyn@action'\MessageBreak
512         for option '#3' in keyval family '#2'%
513       }\@ehc
514     }{%
515       \csname KV0@action@KV0dyn@action\endcsname{#2}{#3}%
516     }%
517   \x
518 }
519 \def\KV0@action@undef#1#2{%
520   \edef\x{\endgroup
521     \ifKV0dyn@global\global\fi
522     \let
523     \expandafter\noexpand\csname KV@#1@#2\endcsname
524     \relax
525     \ifKV0dyn@global\global\fi
526     \let
527     \expandafter\noexpand\csname KV@#1@#2@default\endcsname
528     \relax
529   }%
530   ^^A\PackageInfo{kvoptions}{%
531     ^^A [option] key '#2' of family '#1'\MessageBreak
532     ^^A is disabled (undef, \ifKV0dyn@global\global\else local\fi)%
533   ^^A}%
534 }
535 \def\KV0@action@ignore#1#2{%
```



```

536 \edef\x{\endgroup
537 \ifKV0dyn@global\global\fi
538 \let
539 \expandafter\noexpand\csname KV@#1@#2\endcsname
540 \noexpand\@gobble
541 \ifKV0dyn@global\global\fi
542 \let
543 \expandafter\noexpand\csname KV@#1@#2@default\endcsname
544 \noexpand\@empty
545 }%
546 ^^A\PackageInfo{kvoptions}{%
547 ^^A [option] key '#2' of family '#1'\MessageBreak
548 ^^A is disabled (ignore, \ifKV0dyn@global\global\else local\fi)%
549 ^^A}%
550 }
551 \def\KV0@action@error{%
552 \KV0@do@action{error}%
553 }
554 \def\KV0@action@warning{%
555 \KV0@do@action{warning}%
556 }

#1 error or warning
#2 <family>
#3 <key>
557 \def\KV0@do@action#1#2#3{%
558 \ifx\KV0dyn@name\relax
559 \PackageError{kvoptions}{%
560 Action type '#1' needs package/class name\MessageBreak
561 for key '#3' in family '#2'%
562 }\@ehc
563 \else
564 \edef\x{\endgroup
565 \noexpand\define@key{#2}{#3}[]{%
566 \expandafter\noexpand\csname KV0@disable@#1\endcsname
567 {\KV0dyn@name}\noexpand\KV0dyn@ext{#3}%
568 }%
569 \ifKV0dyn@global
570 \global\let
571 \expandafter\noexpand\csname KV@#2@#3\endcsname
572 \expandafter\noexpand\csname KV@#2@#3\endcsname
573 \global\let
574 \expandafter\noexpand\csname KV@#2@#3@default\endcsname
575 \expandafter\noexpand\csname KV@#2@#3@default\endcsname
576 \fi
577 }%
578 ^^A\ifx\KV0dyn@ext\@clsextension
579 ^^A \expandafter\ClassInfo
580 ^^A\else
581 ^^A \expandafter\PackageInfo
582 ^^A\fi
583 ^^A{\KV0dyn@name}{%
584 ^^A [option] key '#3' of family '#2'\MessageBreak
585 ^^A is disabled (#1, \ifKV0dyn@global\global\else local\fi)%
586 ^^A}%
587 \fi
588 }
589 \def\KV0@disable@error#1#2#3{%
590 \ifx#2\@clsextension

```

```

591 \expandafter\ClassError
592 \else
593 \expandafter\PackageError
594 \fi
595 {#1}{%
596 Option ‘#3’ is given too late,\MessageBreak
597 now the option is ignored%
598 } \@ehc
599 }
600 \def\KV0@disable@warning#1#2#3{%
601 \ifx#2\@clsextension
602 \expandafter\ClassWarning
603 \else
604 \expandafter\PackageWarning
605 \fi
606 {#1}{%
607 Option ‘#3’ is already consumed\MessageBreak
608 and has no effect%
609 }%
610 }

```

## 6.4 Change option code

### 6.4.1 \AddToKeyvalOption

\AddToKeyvalOption

```

611 \newcommand*{\AddToKeyvalOption}{%
612 \@ifstar{%
613 \beginingroup
614 \edef\x{\endgroup
615 \noexpand\KV0@AddToKeyvalOption{\KV0@family}%
616 }%
617 \x
618 }%
619 \KV0@AddToKeyvalOption
620 }

```

\KV0@AddToKeyvalOption

```

621 \def\KV0@AddToKeyvalOption#1#2{%
622 \@ifundefined{KV@#1@#2}{%
623 \PackageWarning{kvoptions}{%
624 Key ‘#2’ of family ‘#1’ does not exist.\MessageBreak
625 Ignoring \string\AddToKeyvalOption
626 }%
627 \@gobble
628 }{%
629 \edef\KV0@next{%
630 \noexpand\KV0@@AddToKeyvalOption
631 \expandafter\noexpand\csname KV@#1@#2\endcsname
632 }%
633 \afterassignment\KV0@next
634 \def\KV0@temp##1%
635 }%
636 }

```

\KV0@@AddToKeyvalOption

```

637 \def\KV0@@AddToKeyvalOption#1{%
638 \beginingroup
639 \toks@\expandafter{#1{##1}}%

```

```

640 \toks@ \expandafter{\the\expandafter\toks@\KV0@temp{##1}}}%
641 \edef\x{\endgroup
642 \noexpand\def\noexpand#1####1{\the\toks@}%
643 }%
644 \x
645 }

```

## 6.5 Process options

### 6.5.1 Get global options

Package xkeyval removes options with equal signs from the global options (`\@classoptionslist`). The effect is that other packages and classes will not see these global options anymore. A bug-report was answered that this behaviour is “by design”. Thus I call it a design bug. Now getting the global options require an algorithm instead of a simple macro call.

```

646 </package>
647 <*package | patch>

```

`\KV0@IfDefThen` Call #2 if command #1 is defined and not `\relax`. (Package `kvoptions-patch` does not load package `ltxcmds`.)

```

648 \def\KV0@IfDefThen#1#2{%
649 \ifx#1\ltx@undefined
650 \else
651 \ifx#1\relax
652 \else
653 #2%
654 \fi
655 \fi
656 }%

```

`\KV0@GetClassOptionsList`

```

657 \def\KV0@GetClassOptionsList{%
658 \let\KV0@classoptionslist\@classoptionslist
659 \KV0@IfDefThen\@classoptionslist{%
660 \KV0@IfDefThen\XKV@documentclass{%
661 \ifx\XKV@documentclass\ltx@empty
662 \else
663 \KV0@IfDefThen\XKV@classoptionslist{%
664 \ifx\XKV@classoptionslist\ltx@empty
665 \else
666 \let\KV0@classoptionslist\XKV@classoptionslist
667 \fi
668 }%
669 \fi
670 }%
671 }%
672 }%

673 </package | patch>
674 <*package>

```

### 6.5.2 \ProcessKeyvalOptions

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

```

675 \newcommand*{\ProcessKeyvalOptions}{%
676 \ifstar{%

```

```

677 \begingroup
678 \edef\x{\endgroup
679 \noexpand\KV@ProcessKeyvalOptions{\KV@family}}%
680 }%
681 \x
682 }%
683 \KV@ProcessKeyvalOptions
684 }

```

```

685 \def\KV@ProcessKeyvalOptions#1{%
686 \let\@tempc\relax
687 \let\KV@temp\@empty

```

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

```

688 \ifx\@currentx\@clsextension
689 \else
690 \KV@GetClassOptionsList
691 \ifx\KV@classoptionslist\relax
692 \else
693 \@for\KV@CurrentOption:=\KV@classoptionslist\do{%
694 \@ifundefined{KV@#1}\expandafter\KV@getkey
695 \KV@CurrentOption=\@nil}{%
696 }{%
697 \@ifundefined{KV@local@#1}\expandafter\KV@getkey
698 \KV@CurrentOption=\@nil}{%
699 \ifx\KV@Patch Y%
700 \edef\KV@temp{%
701 \etex@unexpanded\expandafter{%
702 \KV@temp
703 }%
704 ,%
705 \etex@unexpanded\expandafter{%
706 \KV@CurrentOption
707 }%
708 ,%
709 }%
710 \ltx@onelevel@sanitize\KV@CurrentOption
711 \else
712 \edef\KV@temp{%
713 \KV@temp
714 ,%
715 \KV@CurrentOption
716 ,%
717 }%
718 \fi
719 \@expandtwoargs\@removeelement\KV@CurrentOption
720 \@unusedoptionlist\@unusedoptionlist
721 }{}%
722 }%
723 }%
724 \fi
725 \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.

```

726 \begingroup
727 \toks\tw@{}%

```

```

728 \ifundefined{opt@\@currname.\@currentx}{%
729 \toks@\expandafter{\KV0@temp}}%
730 }{%
731 \toks@\expandafter\expandafter\expandafter{%
732 \csname opt@\@currname.\@currentx\endcsname
733 }%
734 \ifx\@currentx\@clsextension
735 \edef\CurrentOption{\the\toks@}%
736 \toks@\expandafter{\KV0@temp}%
737 \@for\CurrentOption:=\CurrentOption\do{%
738 \ifundefined{%
739 KV@#1@\expandafter\KV0@getkey\CurrentOption=\@nil
740 }{%

```

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

```

741 \ifundefined{KV0@default@\@currname.\@currentx}{%
742 \ifx\KV0@Patch Y%
743 \ltx@onelevel@sanitize\CurrentOption
744 \fi
745 \ifx\@unusedoptionlist\@empty
746 \global\let\@unusedoptionlist\CurrentOption
747 \else
748 \expandafter\expandafter\expandafter\gdef
749 \expandafter\expandafter\expandafter\@unusedoptionlist
750 \expandafter\expandafter\expandafter{%
751 \expandafter\@unusedoptionlist
752 \expandafter,\CurrentOption
753 }%
754 \fi
755 }{%
756 \toks\tw@\expandafter{%
757 \the\toks\expandafter\tw@\expandafter,\CurrentOption
758 }%
759 }%
760 }{%
761 \toks@\expandafter{%
762 \the\expandafter\toks@\expandafter,\CurrentOption
763 }%
764 }%
765 }%
766 \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.

```

767 \ifundefined{KV0@default@\@currname.\@currentx}{%
768 \toks@\expandafter\expandafter\expandafter{%
769 \expandafter\KV0@temp\the\toks@
770 }%
771 }{%
772 \edef\CurrentOption{\the\toks@}%
773 \toks@\expandafter{\KV0@temp}%
774 \@for\CurrentOption:=\CurrentOption\do{%
775 \ifundefined{%
776 KV@#1@\expandafter\KV0@getkey\CurrentOption=\@nil
777 }{%
778 \toks\tw@\expandafter{%
779 \the\toks\expandafter\tw@\expandafter,\CurrentOption

```

```

780         }%
781     }{%
782         \toks@\expandafter{%
783             \the\expandafter\toks@\expandafter,\CurrentOption
784         }%
785     }%
786 }%
787 }%
788 \fi
789 }%
790 \edef\KVO@temp{\endgroup
791     \noexpand\KVO@calldefault{\the\toks\tw@}%
792     \noexpand\KVO@setkeys{#1}{\the\toks@}%
793 }%
794 \KVO@temp

```

Some cleanup of \ProcessOptions.

```

795 \let\CurrentOption\@empty
796 \AtEndOfPackage{\let\@unprocessedoptions\relax}%
797 }

```

### 6.5.3 \ProcessLocalKeyvalOptions

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

```

798 \newcommand*{\ProcessLocalKeyvalOptions}{%
799     \@ifstar{%
800         \begingroup
801             \edef\x{\endgroup
802                 \noexpand\KVO@ProcessLocalKeyvalOptions{\KVO@family}%
803             }%
804         \x
805     }%
806     \KVO@ProcessLocalKeyvalOptions
807 }

808 \def\KVO@ProcessLocalKeyvalOptions#1{%
809     \let\@tempc\relax
810     \let\KVO@temp\@empty

```

Check if \ProcessLocalKeyvalOptions is called inside a package.

```

811 \ifx\@currentx\@pkgextension
812 \else
813     \PackageError{kvoptions}{%
814         \string\ProcessLocalKeyvalOptions\space is intended for packages only%
815     }\@ehc
816 \fi

```

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

```

817 \begingroup
818     \toks\tw@{%
819         \@ifundefined{opt@\@currname.\@currentx}{%
820             \toks@\expandafter{\KVO@temp}%
821         }{%
822             \toks@\expandafter\expandafter\expandafter{%
823                 \csname opt@\@currname.\@currentx\endcsname
824             }%

```

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.

```

825 \ifundefined{KVO@default@\currname.\@currentx}{%
826 \toks@\expandafter\expandafter\expandafter{%
827 \expandafter\KVO@temp\the\toks@
828 }%
829 }%
830 \edef\CurrentOption{\the\toks@}%
831 \toks@\expandafter{\KVO@temp}%
832 \@for\CurrentOption:=\CurrentOption\do{%
833 \ifundefined{%
834 KV@#1@\expandafter\KVO@getkey\CurrentOption=\@nil
835 }{%
836 \toks\tw@\expandafter{%
837 \the\toks\expandafter\tw@\expandafter,\CurrentOption
838 }%
839 }{%
840 \toks@\expandafter{%
841 \the\expandafter\toks@\expandafter,\CurrentOption
842 }%
843 }%
844 }%
845 }%
846 }%
847 \edef\KVO@temp{\endgroup
848 \noexpand\KVO@calldefault{\the\toks\tw@}%
849 \noexpand\KVO@setkeys{#1}{\the\toks@}%
850 }%
851 \KVO@temp

```

Some cleanup of \ProcessOptions.

```

852 \let\CurrentOption\@empty
853 \AtEndOfPackage{\let\unprocessedoptions\relax}%
854 }

```

#### 6.5.4 Helper macros

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

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

`\KVO@calldefault`

```

856 \def\KVO@calldefault#1{%
857 \begingroup
858 \def\x{#1}%
859 \expandafter\endgroup
860 \ifx\x\@empty
861 \else
862 \@for\CurrentOption:=#1\do{%
863 \ifx\CurrentOption\@empty
864 \else
865 \expandafter\KVO@setcurrents\CurrentOption=\@nil
866 \@nameuse{KVO@default@\currname.\@currentx}%
867 \fi
868 }%
869 \fi
870 }

```

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

```

871 \def\KVO@setcurrents#1=#2\@nil{%
872 \def\CurrentOptionValue{#2}%

```

```

873 \ifx\CurrentOptionValue\@empty
874 \let\CurrentOptionKey\CurrentOption
875 \let\CurrentOptionValue\relax
876 \else
877 \edef\CurrentOptionKey{\zap@space#1 \@empty}%
878 \expandafter\KV@setcurrentvalue\CurrentOption\@nil
879 \fi
880 }

```

`\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.

```

881 \def\KV@setcurrentvalue#1=#2\@nil{%
882 \KV@@sp@def\CurrentOptionValue{#2}%
883 }

```

## 6.6 plain T<sub>E</sub>X

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

```

884 \begingroup\expandafter\expandafter\expandafter\endgroup
885 \expandafter\ifx\csname documentclass\endcsname\relax
886 \def\ProcessKeyvalOptions{%
887 \@ifstar{}\@gobble
888 }%
889 \fi

890 \KV@AtEnd%
891 \endpackage

```

## 6.7 Package kvoptions-patch

```

892 (*patch)
893 \NeedsTeXFormat{LaTeX2e}
894 \begingroup\catcode61\catcode48\catcode32=10\relax%
895 \catcode13=5 % ^~M
896 \endlinechar=13 %
897 \catcode123=1 % {
898 \catcode125=2 % }
899 \catcode64=11 % @
900 \def\x{\endgroup
901 \expandafter\edef\csname KV@AtEnd\endcsname{%
902 \endlinechar=\the\endlinechar\relax
903 \catcode13=\the\catcode13\relax
904 \catcode32=\the\catcode32\relax
905 \catcode35=\the\catcode35\relax
906 \catcode61=\the\catcode61\relax
907 \catcode64=\the\catcode64\relax
908 \catcode123=\the\catcode123\relax
909 \catcode125=\the\catcode125\relax
910 }%
911 }%
912 \x\catcode61\catcode48\catcode32=10\relax%
913 \catcode13=5 % ^~M
914 \endlinechar=13 %
915 \catcode35=6 % #
916 \catcode64=11 % @
917 \catcode123=1 % {
918 \catcode125=2 % }
919 \def\TMP@EnsureCode#1#2{%

```



```

920 \edef\KVO@AtEnd{%
921   \KVO@AtEnd
922   \catcode#1=\the\catcode#1\relax
923 }%
924 \catcode#1=#2\relax
925 }
926 \TMP@EnsureCode{39}{12}% '
927 \TMP@EnsureCode{40}{12}% (
928 \TMP@EnsureCode{41}{12}% )
929 \TMP@EnsureCode{43}{12}% +
930 \TMP@EnsureCode{44}{12}% ,
931 \TMP@EnsureCode{45}{12}% -
932 \TMP@EnsureCode{46}{12}% .
933 \TMP@EnsureCode{47}{12}% /
934 \TMP@EnsureCode{58}{12}% :
935 \TMP@EnsureCode{60}{12}% <
936 \TMP@EnsureCode{62}{12}% >
937 \TMP@EnsureCode{91}{12}% [
938 \TMP@EnsureCode{93}{12}% ]
939 \TMP@EnsureCode{96}{12}% '
940 \TMP@EnsureCode{124}{12}% |
941 \edef\KVO@AtEnd{\KVO@AtEnd\noexpand\endinput}
942 \ProvidesPackage{kvoptions-patch}%
943 [2016/05/16 v3.12 LaTeX patch for keyval options (H0)]%

Check for  $\varepsilon$ -TeX.
944 \begingroup\expandafter\expandafter\expandafter\endgroup
945 \expandafter\ifx\csname eTeXversion\endcsname\relax
946   \PackageWarningNoLine{kvoptions-patch}{%
947     Package loading is aborted, because e-TeX is missing%
948   }%
949   \expandafter\KVO@AtEnd
950 \fi%

Package etexcmds for \etex@unexpanded.
951 \RequirePackage{etexcmds}[2007/09/09]
952 \ifetex@unexpanded
953 \else
954   \PackageError{kvoptions-patch}{%
955     Could not find eTeX's \string\unexpanded.\MessageBreak
956     Try adding \string\RequirePackage\string{etexcmds\string} %
957     before \string\documentclass%
958   }\@ehd
959   \expandafter\KVO@AtEnd
960 \fi%

Check for package xkvltxp.
961 \@ifpackageloaded{xkvltxp}{%
962   \PackageWarningNoLine{kvoptions}{%
963     Option 'patch' cannot be used together with\MessageBreak
964     package 'xkvltxp' that is already loaded.\MessageBreak
965     Therefore package loading is aborted%
966   }%
967   \KVO@AtEnd
968 }{%
969 \def\@if@ptions#1#2#3{%
970   \begingroup
971     \KVO@normalize\KVO@temp{#3}%
972     \edef\x{\endgroup
973       \noexpand\@if@ptions{%

```

```

974         \detokenize\expandafter\expandafter\expandafter{%
975         \csname opt@#2.#1\endcsname
976     }%
977 }{%
978     \detokenize\expandafter{\KVO@temp}%
979 }%
980 }%
981 \x
982 }

983 \def\@pass@options#1#2#3{%
984     \KVO@normalize\KVO@temp{#2}%
985     \@ifundefined{opt@#3.#1}{%
986         \expandafter\gdef\csname opt@#3.#1%
987             \expandafter\endcsname\expandafter{%
988                 \KVO@temp
989             }%
990     }{%
991         \expandafter\gdef\csname opt@#3.#1%
992             \expandafter\expandafter\expandafter\endcsname
993             \expandafter\expandafter\expandafter{%
994                 \csname opt@#3.#1\expandafter\endcsname\expandafter,\KVO@temp
995             }%
996     }%
997 }

998 \def\ProcessOptions{%
999     \let\ds@\@empty
1000     \@ifundefined{opt@\@currname.\@current}{%
1001         \let\@curroptions\@empty
1002     }{%
1003         \expandafter\expandafter\expandafter\def
1004         \expandafter\expandafter\expandafter\@curroptions
1005         \expandafter\expandafter\expandafter{%
1006             \csname opt@\@currname.\@current\endcsname
1007         }%
1008     }%
1009     \@ifstar\KVO@xprocess@options\KVO@process@options
1010 }

1011 \def\KVO@process@options{%
1012     \@for\CurrentOption:=\@declaredoptions\do{%
1013         \ifx\CurrentOption\@empty
1014         \else
1015             \begingroup
1016                 \ifx\@current\@clsextension
1017                     \toks@{}%
1018                 \else
1019                     \KVO@GetClassOptionsList
1020                     \toks@\expandafter{\KVO@classoptionslist,}%
1021                 \fi
1022                 \toks\tw@\expandafter{\@curroptions}%
1023                 \edef\x{\endgroup
1024                     \noexpand\in@{,\CurrentOption,}{,\the\toks@\the\toks\tw@,}%
1025                 }%
1026             \x
1027             \ifin@
1028                 \KVO@use@option
1029                 \expandafter\let\csname ds@\CurrentOption\endcsname\@empty
1030             \fi

```

```

1031 \fi
1032 }%
1033 \KVO@process@ptions
1034 }

1035 \def\KVO@xprocess@ptions{%
1036 \ifx\@current\@clsextension
1037 \else
1038 \KVO@GetClassOptionsList
1039 \@for\CurrentOption:=\KVO@classoptionslist\do{%
1040 \ifx\CurrentOption\@empty
1041 \else
1042 \KVO@in@\CurrentOption\@declaredoptions
1043 \ifin@
1044 \KVO@use@option
1045 \expandafter\let\csname ds@\CurrentOption\endcsname\@empty
1046 \fi
1047 \fi
1048 }%
1049 \fi
1050 \KVO@process@ptions
1051 }

1052 \def\KVO@in@#1#2{%
1053 \in@false
1054 \begingroup
1055 \@for\x:=#2\do{%
1056 \ifx\x#1\relax
1057 \in@true
1058 \fi
1059 }%
1060 \edef\x{\endgroup
1061 \ifin@
1062 \noexpand\in@true
1063 \fi
1064 }%
1065 \x
1066 }

1067 \def\KVO@process@ptions{%
1068 \@for\CurrentOption:=\@curroptions\do{%
1069 \@ifundefined{ds@\KVO@SanitizedCurrentOption}{%
1070 \KVO@use@option
1071 \default@ds
1072 }%
1073 \KVO@use@option
1074 }%
1075 \@for\CurrentOption:=\@declaredoptions\do{%
1076 \expandafter\let\csname ds@\CurrentOption\endcsname\relax
1077 }%
1078 \let\CurrentOption\@empty
1079 \let\@fileswith@ptions\@@fileswith@ptions
1080 \AtEndOfPackage{\let\@unprocessedoptions\relax}%
1081 }

1082 \def\KVO@use@option{%
1083 \begingroup
1084 \edef\x{\endgroup
1085 \noexpand\@removeelement{%
1086 \detokenize\expandafter{\CurrentOption}%
1087 }{%

```

```

1088      \detokenize\expandafter{\@unusedoptionlist}%
1089    }%
1090  }%
1091  \x\@unusedoptionlist
1092  \csname ds@\KVO@SanitizedCurrentOption\endcsname
1093 }

1094 \def\OptionNotUsed{%
1095   \ifx\@currentx\@clsextension
1096     \xdef\@unusedoptionlist{%
1097       \ifx\@unusedoptionlist\@empty
1098       \else
1099         \detokenize\expandafter{\@unusedoptionlist,}%
1100       \fi
1101       \detokenize\expandafter{\CurrentOption}%
1102     }%
1103   \fi
1104 }

    Variant of \ExecuteOptions that better protects \CurrentOption.
1105 \def\CurrentOption@SaveLevel{0}
1106 \def\ExecuteOptions{%
1107   \expandafter\KVO@ExecuteOptions
1108     \csname CurrentOption@\CurrentOption@SaveLevel\endcsname
1109 }
1110 \def\KVO@ExecuteOptions#1#2{%
1111   \let#1\CurrentOption
1112   \edef\CurrentOption@SaveLevel{%
1113     \the\numexpr\CurrentOption@SaveLevel+1%
1114   }%
1115   \@for\CurrentOption:=#2\do{%
1116     \csname ds@\CurrentOption\endcsname
1117   }%
1118   \edef\CurrentOption@SaveLevel{%
1119     \the\numexpr\CurrentOption@SaveLevel-1%
1120   }%
1121   \let\CurrentOption#1%
1122 }

1123 \def\KVO@fileswithoptions#1[#2]#3[#4]{%
1124   \ifx#1\@clsextension
1125     \ifx\@classoptionslist\relax
1126       \KVO@normalize\KVO@temp{#2}%
1127       \expandafter\gdef\expandafter\@classoptionslist\expandafter{%
1128         \KVO@temp
1129       }%
1130       \def\reserved@a{%
1131         \KVO@onefilewithoptions{#3}[#{#2}][#{#4}]#1%
1132         \@documentclasshook
1133       }%
1134     \else
1135       \def\reserved@a{%
1136         \KVO@onefilewithoptions{#3}[#{#2}][#{#4}]#1%
1137       }%
1138     \fi
1139   \else
1140     \begingroup
1141     \let\KVO@temp\relax
1142     \let\KVO@onefilewithoptions\relax
1143     \let\@pkgextension\relax

```

```

1144 \def\reserved@b##1,{%
1145 \ifx\@nil##1\relax
1146 \else
1147 \ifx\relax##1\relax
1148 \else
1149 \KV@onefilewithoptions{##1}{\KV@temp}{\#4}%
1150 \@pkgextension
1151 \fi
1152 \expandafter\reserved@b
1153 \fi
1154 }%
1155 \edef\reserved@a{\zap@space#3 \empty}%
1156 \edef\reserved@a{\expandafter\reserved@b\reserved@a,\@nil,}%
1157 \toks@{\#2}%
1158 \def\KV@temp{\the\toks@}%
1159 \edef\reserved@a{\endgroup \reserved@a}%
1160 \fi
1161 \reserved@a
1162 }

1163 \def\KV@onefilewithoptions#1[#2][#3]#4{%
1164 \@pushfilename
1165 \xdef\@currname{#1}%
1166 \global\let\@currentx#4%
1167 \expandafter\let\csname\@currname.\@currentx-h@k\endcsname\empty
1168 \let\CurrentOption\empty
1169 \@reset@ptions
1170 \makeatletter
1171 \def\reserved@a{%
1172 \@ifl@aded\@currentx{#1}{%
1173 \@if@ptions\@currentx{#1}{#2}{%
1174 }{%
1175 \begingroup
1176 \@ifundefined{opt@#1.\@currentx}{%
1177 \def\x{%
1178 }{%
1179 \edef\x{%
1180 \expandafter\expandafter\expandafter\strip@prefix
1181 \expandafter\meaning\csname opt@#1.\@currentx\endcsname
1182 }%
1183 }%
1184 \def\y{#2}%
1185 \edef\y{\expandafter\strip@prefix\meaning\y}%
1186 \@latex@error{Option clash for \cls@pkg\space #1}{%
1187 The package #1 has already been loaded %
1188 with options:\MessageBreak
1189 \space\space[\x]\MessageBreak
1190 There has now been an attempt to load it %
1191 with options:\MessageBreak
1192 \space\space[\y]\MessageBreak
1193 Adding the global options:\MessageBreak
1194 \space\space
1195 \x,\y\MessageBreak
1196 to your \noexpand\documentclass declaration may fix this.%
1197 \MessageBreak
1198 Try typing \space <return> \space to proceed.%
1199 }%
1200 \endgroup
1201 }%

```

```

1202 }{%
1203   \@passoptions\@currentx{#2}{#1}%
1204   \global\expandafter
1205   \let\csname ver@\@currname.\@currentx\endcsname\@empty
1206   \InputIfFileExists
1207     {\@currname.\@currentx}%
1208     {}%
1209     {\@missingfileerror\@currname\@currentx}%
1210   \let\@unprocessedoptions\@@unprocessedoptions
1211   \csname\@currname.\@currentx-h@@k\endcsname
1212   \expandafter\let\csname\@currname.\@currentx-h@@k\endcsname
1213     \@undefined
1214   \@unprocessedoptions
1215 }%
1216 \@ifl@ter\@currentx{#1}{#3}{%
1217 }{%
1218   \@latex@warning@no@line{%
1219     You have requested,\on@line, %
1220     version\MessageBreak
1221     #3' of \@cls@pkg\space #1,\MessageBreak
1222     but only version\MessageBreak
1223     '\csname ver@#1.\@currentx\endcsname'\MessageBreak
1224     is available%
1225   }%
1226 }%
1227 \ifx\@currentx\@clsextension\let\LoadClass\@twoloadclasserror\fi
1228 \@popfilename
1229 \@reset@options
1230 }%
1231 \reserved@a
1232 }

1233 \def\@unknownoptionerror{%
1234   \@latex@error{%
1235     Unknown option '\KVO@SanitizedCurrentOption' %
1236     for \@cls@pkg\space'\@currname'%
1237   }{%
1238     The option '\KVO@SanitizedCurrentOption' was not declared in %
1239     \@cls@pkg\space'\@currname', perhaps you\MessageBreak
1240     misspelled its name. %
1241     Try typing \space <return> %
1242     \space to proceed.%
1243   }%
1244 }

1245 \def\@@unprocessedoptions{%
1246   \ifx\@currentx\@pkgextension
1247     \@ifundefined{opt@\@currname.\@currentx}{%
1248       \let\@curroptions\@empty
1249     }{%
1250       \expandafter\let\expandafter\@curroptions
1251         \csname opt@\@currname.\@currentx\endcsname
1252     }%
1253     \@for\CurrentOption:=\@curroptions\do{%
1254       \ifx\CurrentOption\@empty\else\@unknownoptionerror\fi
1255     }%
1256   \fi
1257 }

1258 \def\KVO@SanitizedCurrentOption{%

```

```

1259 \expandafter\strip@prefix\meaning\CurrentOption
1260 }

      Normalize option list.
1261 \def\KVO@normalize#1#2{%
1262   \let\KVO@result\@empty
1263   \KVO@splitcomma#2,\@nil
1264   \let#1\KVO@result
1265 }
1266 \def\KVO@splitcomma#1,#2\@nil{%
1267   \KVO@ifempty{#1}{}%
1268   \KVO@checkkv#1=\@nil
1269   }%
1270   \KVO@ifempty{#2}{%\KVO@splitcomma#2\@nil}%
1271 }
1272 \def\KVO@ifempty#1{%
1273   \expandafter\ifx\expandafter\\\detokenize{#1}\\\%
1274     \expandafter\@firstoftwo
1275   \else
1276     \expandafter\@secondoftwo
1277   \fi
1278 }
1279 \def\KVO@checkkv#1=#2\@nil{%
1280   \KVO@ifempty{#2}{%
1281     % option without value
1282     \edef\KVO@x{\zap@space#1 \@empty}%
1283     \ifx\KVO@x\@empty
1284       % ignore empty option
1285     \else
1286       % append to list
1287       \edef\KVO@result{%
1288         \etex@unexpanded\expandafter{\KVO@result},\KVO@x
1289       }%
1290     \fi
1291   }{%
1292     % #1: "key", #2: "value="
1293     % add key part
1294     \edef\KVO@result{%
1295       \etex@unexpanded\expandafter{\KVO@result},%
1296       \zap@space#1 \@empty
1297     }%
1298     \futurelet\@let@token\KVO@checkfirsttok#2 \@nil| = \@nil|\KVO@nil
1299   }%
1300 }
1301 \def\KVO@checkfirsttok{%
1302   \ifx\@let@token\bgroup
1303     % no space at start
1304     \expandafter\KVO@removelastspace\expandafter=%
1305     % "<value><spaceopt>= \@nil"
1306   \else
1307     \expandafter\KVO@checkfirstA
1308   \fi
1309 }
1310 \def\KVO@checkfirstA#1 #2\@nil{%
1311   \KVO@ifempty{#2}{%
1312     \KVO@removelastspace=#1 \@nil
1313   }{%
1314     \KVO@ifempty{#1}{%
1315       \KVO@removelastspace=#2\@nil

```

```

1316 }{%
1317 \KV0@removelastspace=#1 #2\@nil
1318 }%
1319 }%
1320 }
1321 \def\KV0@removelastspace#1 = \@nil|#2\KV0@nil{%
1322 \KV0@ifempty{#2}{%
1323 \edef\KV0@result{%
1324 \etex@unexpanded\expandafter{\KV0@result}%
1325 \etex@unexpanded\expandafter{\KV0@removegarbage#1\KV0@nil}%
1326 }%
1327 }{%
1328 \edef\KV0@result{%
1329 \etex@unexpanded\expandafter{\KV0@result}%
1330 \etex@unexpanded{#1}%
1331 }%
1332 }%
1333 }
1334 \def\KV0@removegarbage#1= \@nil#2\KV0@nil{#1}%
Arguments #1 and #2 are macros.
1335 \def\KV0@removeelement#1#2{%
1336 \begingroup
1337 \toks@={}%
1338 \@for\x:=#2\do{%
1339 \ifx\x\@empty
1340 \else
1341 \ifx\x#1\relax
1342 \else
1343 \edef\t{\the\toks@}%
1344 \ifx\t\@empty
1345 \else
1346 \toks@\expandafter{\the\toks@,}%
1347 \fi
1348 \toks@\expandafter{\the\expandafter\toks@\x}%
1349 \fi
1350 \fi
1351 }%
1352 \edef\x{\endgroup
1353 \def\noexpand#2{\the\toks@}%
1354 }%
1355 \x
1356 }
1357 \let\@@fileswith@pti@ns\KV0@fileswith@pti@ns
1358 \ifx\@@fileswith@pti@ns\@badrequireerror
1359 \else
1360 \let\@@fileswith@pti@ns\KV0@fileswith@pti@ns
1361 \fi

```

\KV0@Patch

```

1362 \let\KV0@Patch=Y
1363 \KV0@AtEnd%
1364 \patch

```

## 7 Test

### 7.1 Preface for standard catcode check



```

1365 <*test1>
1366 \input miniltx.tex\relax
1367 </test1>

```

## 7.2 Catcode checks for loading

```

1368 <*test1>
1369 \catcode'\{=1 %
1370 \catcode'\}=2 %
1371 \catcode'\#=6 %
1372 \catcode'\@=11 %
1373 \expandafter\ifx\csname count@\endcsname\relax
1374   \countdef\count@=255 %
1375 \fi
1376 \expandafter\ifx\csname @gobble\endcsname\relax
1377   \long\def\@gobble#1{}%
1378 \fi
1379 \expandafter\ifx\csname @firstofone\endcsname\relax
1380   \long\def\@firstofone#1{#1}%
1381 \fi
1382 \expandafter\ifx\csname loop\endcsname\relax
1383   \expandafter\@firstofone
1384 \else
1385   \expandafter\@gobble
1386 \fi
1387 {%
1388   \def\loop#1\repeat{%
1389     \def\body{#1}%
1390     \iterate
1391   }%
1392   \def\iterate{%
1393     \body
1394     \let\next\iterate
1395   \else
1396     \let\next\relax
1397   \fi
1398   \next
1399 }%
1400 \let\repeat=\fi
1401 }%
1402 \def\RestoreCatcodes{}
1403 \count@=0 %
1404 \loop
1405   \edef\RestoreCatcodes{%
1406     \RestoreCatcodes
1407     \catcode\the\count@=\the\catcode\count@\relax
1408   }%
1409   \ifnum\count@<255 %
1410     \advance\count@ 1 %
1411   \repeat
1412
1413 \def\RangeCatcodeInvalid#1#2{%
1414   \count@=#1\relax
1415   \loop
1416     \catcode\count@=15 %
1417     \ifnum\count@<#2\relax
1418       \advance\count@ 1 %
1419     \repeat
1420 }

```

```

1421 \def\RangeCatcodeCheck#1#2#3{%
1422   \count@=#1\relax
1423   \loop
1424     \ifnum#3=\catcode\count@
1425     \else
1426       \errmessage{%
1427         Character \the\count@\space
1428         with wrong catcode \the\catcode\count@\space
1429         instead of \number#3%
1430       }%
1431     \fi
1432   \ifnum\count@<#2\relax
1433     \advance\count@ 1 %
1434   \repeat
1435 }
1436 \def\space{ }
1437 \expandafter\ifx\csname LoadCommand\endcsname\relax
1438   \def\LoadCommand{\input kvoptions.sty\relax}%
1439 \fi
1440 \def\Test{%
1441   \RangeCatcodeInvalid{0}{47}%
1442   \RangeCatcodeInvalid{58}{64}%
1443   \RangeCatcodeInvalid{91}{96}%
1444   \RangeCatcodeInvalid{123}{255}%
1445   \catcode'\@=12 %
1446   \catcode'\=0 %
1447   \catcode'\%=14 %
1448   \LoadCommand
1449   \RangeCatcodeCheck{0}{36}{15}%
1450   \RangeCatcodeCheck{37}{37}{14}%
1451   \RangeCatcodeCheck{38}{47}{15}%
1452   \RangeCatcodeCheck{48}{57}{12}%
1453   \RangeCatcodeCheck{58}{63}{15}%
1454   \RangeCatcodeCheck{64}{64}{12}%
1455   \RangeCatcodeCheck{65}{90}{11}%
1456   \RangeCatcodeCheck{91}{91}{15}%
1457   \RangeCatcodeCheck{92}{92}{0}%
1458   \RangeCatcodeCheck{93}{96}{15}%
1459   \RangeCatcodeCheck{97}{122}{11}%
1460   \RangeCatcodeCheck{123}{255}{15}%
1461   \RestoreCatcodes
1462 }
1463 \Test
1464 \csname @@end\endcsname
1465 \end
1466 \</test1>
1467 \<*/test2>
1468 \NeedsTeXFormat{LaTeX2e}
1469 \makeatletter
1470 \catcode'\@=11 %
1471 \def\RestoreCatcodes{}
1472 \count@=0 %
1473 \loop
1474   \edef\RestoreCatcodes{%
1475     \RestoreCatcodes
1476     \catcode\the\count@=\the\catcode\count@\relax
1477   }%
1478   \ifnum\count@<255 %

```

```

1479 \advance\count@\@ne
1480 \repeat
1481
1482 \def\RangeCatcodeInvalid#1#2{%
1483   \count@=#1\relax
1484   \loop
1485     \catcode\count@=15 %
1486     \ifnum\count@<#2\relax
1487       \advance\count@\@ne
1488     \repeat
1489 }
1490 \def\Test#1{%
1491   \RangeCatcodeInvalid{0}{47}%
1492   \RangeCatcodeInvalid{58}{64}%
1493   \RangeCatcodeInvalid{91}{96}%
1494   \RangeCatcodeInvalid{123}{255}%
1495   \catcode'\@=12 %
1496   \catcode'\=0 %
1497   \catcode'\{=1 %
1498   \catcode'\}=2 %
1499   \catcode'\#=6 %
1500   \catcode'\[=12 %
1501   \catcode'\]=12 %
1502   \catcode'\%=14 %
1503   \catcode'\ =10 %
1504   \catcode13=5 %
1505   #1\relax
1506   \RestoreCatcodes
1507 }
1508 \Test{\RequirePackage{kvoptions-patch}}%
1509 \Test{\RequirePackage{kvoptions}}%
1510 \csname @@end\endcsname
1511 \end{test2}

1512 \begin{test3}
1513 \NeedsTeXFormat{LaTeX2e}
1514 \makeatletter
1515 \RequirePackage{kvoptions}[2016/05/16]
1516 \def\msg#1{\immediate\write16}
1517 \define@key{testfamily}{testkey}{%
1518   \msg{[testfamily/testkey/#1]}%
1519 }
1520 \define@key{testfamily}{testdefaultkey}[testdefault]{%
1521   \msg{[testfamily/testdefaultkey/#1]}%
1522 }
1523 \AddToKeyvalOption{testfamily}{testkey}{%
1524   \msg{[addition/#1]}%
1525 }
1526 \AddToKeyvalOption{testfamily}{testdefaultkey}{%
1527   \msg{[addition/#1]}%
1528 }
1529 \setkeys{testfamily}{%
1530   testkey=testA,%
1531   testdefaultkey=testB,%
1532   testdefaultkey,%
1533 }
1534 \SetupKeyvalOptions{%
1535   family=testfamily%
1536 }

```

```

1537 \AddToKeyvalOption*{testkey}{%
1538   \msg{[star addition/#1]}%
1539 }
1540 \AddToKeyvalOption*{testdefaultkey}{%
1541   \msg{[star addition/#1]}%
1542 }
1543 \setkeys{testfamily}{%
1544   testkey=testA,%
1545   testdefaultkey=testB,%
1546   testdefaultkey,%
1547 }
1548 \end
1549 \test3}

1550 (*test4pkg)
1551 \NeedsTeXFormat{LaTeX2e}
1552 \ProvidesPackage{kvoptions-test4}[2016/05/16 package for testing]
1553 \RequirePackage{kvoptions}[2016/05/16]
1554 \SetupKeyvalOptions{%
1555   family=F00,%
1556   prefix=foo,%
1557   setkeys=\kvsetkeys,%
1558 }
1559 \DeclareStringOption{str}
1560 \define@key{F00}{set}{%
1561   \setkeys{BAR}{strbar={#1}}%
1562 }
1563 \define@key{BAR}{strbar}{%
1564   \def\foostr{[BAR:#1]}%
1565 }
1566 \ProcessKeyvalOptions*
1567 \test4pkg)

1568 (*test4)
1569 \NeedsTeXFormat{LaTeX2e}
1570 \ProvidesFile{kvoptions-test4.tex}[2016/05/16 test file]
1571 \RequirePackage[%
1572   str=A,set=B,str=C,%
1573 ]{kvoptions-test4}[2016/05/16]
1574 \def\TestExpected{C}
1575 \ifx\foostr\TestExpected
1576   \typeout{* Test ok.}%
1577 \else
1578   \typeout{* Result: [\foostr]}%
1579   \typeout{* Expected: [\TestExpected]}%
1580   \errmessage{Test failed!}%
1581 \fi
1582 \csname @@end\endcsname\end
1583 \test4}

```

## 8 Installation

### 8.1 Download

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

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

[CTAN:macros/latex/contrib/oberdiek/kvoptions.pdf](http://ctan.org/macros/latex/contrib/oberdiek/kvoptions.pdf) Documentation.

---

<sup>1</sup><http://ctan.org/pkg/kvoptions>

**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>test/kvoptions-test3.tex</code>	→ <code>doc/latex/oberdiek/test/kvoptions-test3.tex</code>
<code>test/kvoptions-test4.tex</code>	→ <code>doc/latex/oberdiek/test/kvoptions-test4.tex</code>
<code>test/kvoptions-test4.sty</code>	→ <code>doc/latex/oberdiek/test/kvoptions-test4.sty</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  $\text{\LaTeX}$ .** The `.dtx` chooses its action depending on the format:

**plain  $\text{\TeX}$ :** Run `docstrip` and extract the files.

**$\text{\LaTeX}$ :** Generate the documentation.

If you insist on using  $\text{\LaTeX}$  for `docstrip` (really, `docstrip` does not need  $\text{\LaTeX}$ ), 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 $\text{\LaTeX}$` :

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

## 9 Catalogue

The following XML file can be used as source for the  [\$\text{\TeX}\$  Catalogue](#). The elements `caption` and `description` are imported from the original XML file from the Catalogue. The name of the XML file in the Catalogue is `kvoptions.xml`.

```
1584 (*catalogue)
1585 <?xml version='1.0' encoding='us-ascii'?>
1586 <!DOCTYPE entry SYSTEM 'catalogue.dtd'>
1587 <entry datestamp='$Date$' modifier='$Author$' id='kvoptions'>
1588   <name>kvoptions</name>
1589   <caption>Key value format for package options.</caption>
1590   <authorref id='auth:oberdiek'>
1591     <copyright owner='Heiko Oberdiek' year='2004,2006,2007,2009-2011'>
1592       <license type='lppl1.3'>
1593         <version number='3.12'>
1594           <description>
1595             This package offers support for package authors who want to
1596             use options in key-value format for their package options.
1597           </p>
1598           The package is part of the <xref refid='oberdiek'>oberdiek</xref> bundle.
1599         </description>
1600       <documentation details='Package documentation'
```

```

1601      href='ctan:/macros/latex/contrib/oberdiek/kvoptions.pdf'/>
1602 <ctan file='true' path='/macros/latex/contrib/oberdiek/kvoptions.dtx'/>
1603 <miktex location='oberdiek'/>
1604 <texlive location='oberdiek'/>
1605 <install path='/macros/latex/contrib/oberdiek/oberdiek.tds.zip'/>
1606 </entry>
1607 </catalogue>

```

## 10 References

- [1] A guide to key-value methods, Joseph Wright, second draft for TUG-Boat, 2009-03-17. <http://www.texdev.net/wp-content/uploads/2009/03/keyval.pdf>
- [2] Package ifthen, David Carlisle, 2001/05/26. [CTAN:macros/latex/base/ifthen.dtx](#)
- [3] Package helvet, Sebastian Rahtz, Walter Schmidt, 2004/01/26. [CTAN:macros/latex/required/psnfss/psfonts.dtx](#)
- [4] Package hyperref, Sebastian Rahtz, Heiko Oberdiek, 2006/02/12. [CTAN:macros/latex/contrib/hyperref/](#)
- [5] Package keyval, David Carlisle, 1999/03/16. [CTAN:macros/latex/required/graphics/keyval.dtx](#)
- [6] Package multicol, Frank Mittelbach, 2004/02/14. [CTAN:macros/latex/required/tools/multicol.dtx](#)
- [7] Package tabularx, David Carlisle, 1999/01/07. [CTAN:macros/latex/required/tools/tabularx.dtx](#)
- [8] Package tracefnt, Frank Mittelbach, Rainer Schöpf, 1997/05/29. [CTAN:macros/latex/base/ltfsstrc.dtx](#)
- [9] Package xkeyval, Hendri Adriaens, 2005/05/07. [CTAN:macros/latex/contrib/xkeyval/](#)
- [10] The 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](#)

## 11 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.

[2009/04/10 v3.1]

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

[2009/07/17 v3.2]

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

[2009/07/21 v3.3]

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

[2009/08/13 v3.4]

- Documentation addition: recommendation for Joseph Wright's review article.
- Documentation addition: local/global options.

[2009/12/04 v3.5]

- `\AddToKeyvalOption` added.

[2009/12/08 v3.6]

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

[2010/02/22 v3.7]

- Missing space in error message added.

[2010/07/23 v3.8]

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

[2010/12/02 v3.9]

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

[2010/12/23 v3.10]

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

[2011/06/30 v3.11]

- Fix because of design bug in package xkeyval that removes global options with equal signs.

[2016/05/16 v3.12]

- Documentation updates.

## 12 Index

Numbers written in *italic* refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; plain numbers refer to the code lines where the entry is used.

Symbols	
<code>\#</code> .....	1371, 1499
<code>\%</code> .....	1447, 1502
<code>\@</code> .....	1372, 1445, 1470, 1495
<code>\@@end</code> .....	1548
<code>\@@fileswith@pti@ns</code> .....	1079, 1357
<code>\@unprocessedoptions</code> ...	1210, 1245
<code>\@SetupDriver</code> .....	41, 43
<code>\@badrequireerror</code> .....	1358
<code>\@classoptionslist</code> 658, 659, 1125, 1127	
<code>\@cls@pkg</code> .....	1186, 1221, 1236, 1239
<code>\@clsextension</code> .....	
.....	295, 296, 315, 335, 336,
	360, 373, 406, 434, 435, 461,
	471, 498, 578, 590, 601, 688,
	734, 1016, 1036, 1095, 1124, 1227
<code>\@current</code> 231, 242, 245, 248, 253, 256,	
	259, 264, 267, 270, 295, 315,
	316, 335, 406, 434, 479, 688,
	728, 732, 734, 741, 767, 811,
	819, 823, 825, 866, 1000, 1006,
	1016, 1036, 1095, 1166, 1167,
	1172, 1173, 1176, 1181, 1203,
	1205, 1207, 1209, 1211, 1212,
	1216, 1223, 1227, 1246, 1247, 1251
<code>\@currname</code> . 57, 228, 242, 245, 246,	
	248, 253, 256, 257, 259, 264,
	267, 270, 294, 316, 334, 411,
	433, 479, 728, 732, 741, 767,
	819, 823, 825, 866, 1000, 1006,
	1165, 1167, 1205, 1207, 1209,
	1211, 1212, 1236, 1239, 1247, 1251
<code>\@curroptions</code> .....	1001,
	1004, 1022, 1068, 1248, 1250, 1253
<code>\@declaredoptions</code> ..	1012, 1042, 1075
<code>\@documentclasshook</code> .....	1132
<code>\@ehc</code> .....	322, 513, 562, 598, 815
<code>\@ehd</code> .....	958
<code>\@empty</code> .....	88, 89, 495, 544, 687,
	745, 795, 810, 852, 860, 863,
	873, 877, 999, 1001, 1013, 1029,
	1040, 1045, 1078, 1097, 1155,
	1167, 1168, 1205, 1248, 1254,
	1262, 1282, 1283, 1296, 1339, 1344
<code>\@expandtwoargs</code> .....	719
<code>\@fileswith@pti@ns</code> ..	1079, 1358, 1360
<code>\@firstofone</code> . 355, 358, 426, 1380, 1383	
<code>\@firstoftwo</code> .....	1274
<code>\@for</code> .....	693,
	737, 774, 832, 862, 1012, 1039,
	1055, 1068, 1075, 1115, 1253, 1338
<code>\@gobble</code> 369, 540, 627, 887, 1377, 1385	
<code>\@gobbletwo</code> .....	424
<code>\@if@pti@ns</code> .....	973
<code>\@if@ptions</code> .....	969, 1173
<code>\@ifdefinable</code> .....	349
<code>\@ifl@aded</code> .....	1172
<code>\@ifl@ter</code> .....	1216
<code>\@ifnextchar</code> .....	387
<code>\@ifpackageloaded</code> .....	961
<code>\@ifstar</code> .....	612, 676, 799, 887, 1009
<code>\@ifundefined</code> .....	
	. 216, 221, 227, 230, 245, 282,
	309, 508, 622, 694, 697, 728,
	738, 741, 767, 775, 819, 825,
	833, 985, 1000, 1069, 1176, 1247
<code>\@latex@error</code> .....	1186, 1234
<code>\@latex@warning@no@line</code> .....	1218
<code>\@let@token</code> .....	1298, 1302
<code>\@missingfileerror</code> .....	1209
<code>\@namedef</code> .....	395, 479
<code>\@nameuse</code> .....	866
<code>\@ne</code> .....	233, 1479, 1487
<code>\@nil</code> .....	695, 698, 739,
	776, 834, 855, 865, 871, 878,
	881, 1145, 1156, 1263, 1266,
	1268, 1270, 1279, 1298, 1305,
	1310, 1312, 1315, 1317, 1321, 1334
<code>\@pass@ptions</code> .....	983, 1203
<code>\@pkgextension</code> .....	298,
	338, 437, 502, 811, 1143, 1150, 1246
<code>\@popfilename</code> .....	1228
<code>\@pushfilename</code> .....	1164
<code>\@removeelement</code> .....	719, 1085

\@reset@ptions	1169, 1229	\countdef	1374
\@secondoftwo	1276	\csname	111, 118, 147, 163, 173, 241, 248, 252, 259, 263, 270, 280, 281, 289, 327, 328, 329, 330, 349, 379, 416, 441, 449, 485, 515, 523, 527, 539, 543, 566, 571, 572, 574, 575, 631, 732, 823, 885, 901, 945, 975, 986, 991, 994, 1006, 1029, 1045, 1076, 1092, 1108, 1116, 1167, 1181, 1205, 1211, 1212, 1223, 1251, 1373, 1376, 1379, 1382, 1437, 1464, 1510, 1582
\@tempc	686, 809	\CurrentOption	35, 37, 38, 41, 58, 62, 64, 455, 735, 737, 739, 743, 746, 752, 757, 762, 772, 774, 776, 779, 783, 795, 830, 832, 834, 837, 841, 852, 862, 863, 865, 874, 878, 1012, 1013, 1024, 1029, 1039, 1040, 1042, 1045, 1068, 1075, 1076, 1078, 1086, 1101, 1111, 1115, 1116, 1121, 1168, 1253, 1254, 1259
\@twoloadclasserror	1227	\CurrentOption@SaveLevel	1105, 1108, 1112, 1113, 1118, 1119
\@typeset@protect	225	\CurrentOptionKey	874, 877
\@undefined	155, 1213	\CurrentOptionValue	56, 872, 873, 875, 882
\@unknownoptionerror	68, 1233, 1254		
\@unprocessedoptions	796, 853, 1080, 1210, 1214		
\@unusedoptionlist	720, 745, 746, 749, 751, 1088, 1091, 1096, 1097, 1099		
\@x@protect	222		
\[	1500		
\]	397, 1273, 1446, 1496		
\{	1369, 1497		
\}	1370, 1498		
\]	1501		
\_	1503		
<b>A</b>			
\AddToKeyvalOption	9, 611, 625, 1523, 1526, 1537, 1540		
\advance	1410, 1418, 1433, 1479, 1487		
\afterassignment	633		
\aftergroup	126		
\AtEndOfPackage	235, 796, 853, 1080		
<b>B</b>			
\body	1389, 1393		
<b>C</b>			
\catcode	99, 100, 102, 103, 104, 105, 106, 107, 108, 109, 110, 130, 131, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 166, 167, 169, 170, 171, 175, 176, 177, 178, 179, 180, 181, 184, 185, 187, 188, 189, 190, 194, 196, 233, 894, 895, 897, 898, 899, 903, 904, 905, 906, 907, 908, 909, 912, 913, 915, 916, 917, 918, 922, 924, 1369, 1370, 1371, 1372, 1407, 1416, 1424, 1428, 1445, 1446, 1447, 1470, 1476, 1485, 1495, 1496, 1497, 1498, 1499, 1500, 1501, 1502, 1503, 1504		
\ClassError	591		
\ClassInfo	374, 407, 472, 579		
\ClassWarning	361, 462, 602		
\comma@parse	482		
\count@	1374, 1403, 1407, 1409, 1410, 1414, 1416, 1417, 1418, 1422, 1424, 1427, 1428, 1432, 1433, 1472, 1476, 1478, 1479, 1483, 1485, 1486, 1487		
		\DeclareBoolOption	6, 20, 22, 276, 491
		\DeclareComplementaryOption	7, 308, 492
		\DeclareDefaultOption	7, 55, 478
		\DeclareLocalOption	8
		\DeclareLocalOptions	481
		\DeclareOption	233, 234
		\DeclareStringOption	5, 26, 28, 50, 52, 386, 493, 1559
		\DeclareVoidOption	7, 32, 33, 34, 422
		\default@ds	1071
		\define@key	240, 251, 262, 293, 333, 403, 432, 496, 500, 565, 1517, 1520, 1560, 1563
		\detokenize	974, 978, 1086, 1088, 1099, 1101, 1273
		\DisableKeyvalOption	8, 504
		\do	693, 737, 774, 832, 862, 1012, 1039, 1055, 1068, 1075, 1115, 1253, 1338
		\documentclass	957, 1196
		\ds@	999
<b>E</b>			
		\emph	48, 87, 91, 93
		\empty	114, 115
		\end	1465, 1582
		\endcsname	111, 118, 147, 163, 173, 242, 248, 253, 259,

264, 270, 280, 281, 289, 327, 328, 329, 330, 349, 379, 416, 441, 449, 485, 515, 523, 527, 539, 543, 566, 571, 572, 574, 575, 631, 732, 823, 885, 901, 945, 975, 987, 992, 994, 1006, 1029, 1045, 1076, 1092, 1108, 1116, 1167, 1181, 1205, 1211, 1212, 1223, 1251, 1373, 1376, 1379, 1382, 1437, 1464, 1510, 1582	\endinput . . . . . 126, 215, 941	\KVO@action@undef . . . . . 519
\endlinechar . . . . . 101, 132, 168, 174, 186, 896, 902, 914	\errmessage . . . . . 1426, 1580	\KVO@action@warning . . . . . 554
\etex@unexpanded . . . . . 701, 705, 1288, 1295, 1324, 1325, 1329, 1330	\ExecuteOptions . . . . . 1106	\KVO@AddToKeyvalOption 615, 619, <u>621</u>
<b>F</b>		\KVO@AtEnd . . . . 192, 193, 215, 890, 920, 921, 941, 949, 959, 967, 1363
\foostr . . . . . 1564, 1575, 1578		\KVO@boolkey . . . . . 294, 334, <u>351</u>
\futurelet . . . . . 1298		\KVO@calldefault . . . . 791, 848, <u>856</u>
<b>G</b>		\KVO@checkfirstA . . . . . 1307, 1310
\gdef . . . . . 748, 986, 991, 1127		\KVO@checkfirsttok . . . . 1298, 1301
<b>I</b>		\KVO@checkkv . . . . . 1268, 1279
\ifetex@unexpanded . . . . . 952		\KVO@classoptionslist . . . . . . . . 658, 666, 691, 693, 1020, 1039
\ifin@ . . . . . 1027, 1043, 1061		\KVO@CurrentOption . . . . . . 693, 695, 698, 706, 710, 715, 719
\ifKVOdyn@global . . . . . 521, 525, 532, 537, 541, 548, 569, 585		\KVO@DeclareLocalOption . . . 482, <u>484</u>
\ifMCS@print . . . . . 79		\KVO@DeclareStringOption 388, 390, <u>393</u>
\ifnum 1409, 1417, 1424, 1432, 1478, 1486		\KVO@disable@error . . . . . 589
\ifpdf . . . . . 25		\KVO@disable@warning . . . . . 600
\ifx . . . . 56, 88, 112, 115, 118, 147, 155, 158, 295, 315, 335, 354, 357, 360, 373, 397, 406, 434, 459, 461, 471, 558, 578, 590, 601, 649, 651, 661, 664, 688, 691, 699, 734, 742, 745, 811, 860, 863, 873, 885, 945, 1013, 1016, 1036, 1040, 1056, 1095, 1097, 1124, 1125, 1145, 1147, 1227, 1246, 1254, 1273, 1283, 1302, 1339, 1341, 1344, 1358, 1373, 1376, 1379, 1382, 1437, 1575		\KVO@do@action . . . . . 552, 555, 557
\immediate . . . . . 120, 149, 1516		\KVO@ExecuteOptions . . . . 1107, 1110
\in@ . . . . . 1024		\KVO@false . . . . . 357, <u>382</u>
\in@false . . . . . 1053		\KVO@family . . . . . <u>240</u> , 293, 317, 333, 403, 432, 485, 615, 679, 802
\in@true . . . . . 1057, 1062		\KVO@fileswith@pti@ns 1123, 1357, 1360
\input . . . . . 1366, 1438		\KVO@GetClassOptionsList . . . . . . . . . . 657, 690, 1019, 1038
\InputIfFileExists . . . . . 1206		\KVO@getkey 694, 697, 739, 776, 834, <u>855</u>
\iterate . . . . . 1390, 1392, 1394		\KVO@ifdefinable . . . . . 277, 278, 279, 325, 326, <u>348</u> , 394, 425
<b>K</b>		\KVO@ifDefThen . . . . <u>648</u> , 659, 660, 663
\KV@sp@def . . . . . 882		\KVO@ifempty . . . . . 1267, 1270, 1272, 1280, 1311, 1314, 1322
\KV@setcurrentvalue . . . . . <u>881</u>		\KVO@in@ . . . . . 1042, 1052
\KVO@AddToKeyvalOption . . . 630, <u>637</u>		\KVO@next . . . . . 629, 633
\KVO@action@error . . . . . 551		\KVO@nil . . . . . 1298, 1321, 1325, 1334
\KVO@action@ignore . . . . . 535		\KVO@normalize . . 971, 984, 1126, 1261
		\KVO@onefilewithoptions . . . . . . . . . 1131, 1136, 1142, 1149, 1163
		\KVO@param . . . . . . 352, 353, 354, 357, 366, 378, 379
		\KVO@Patch . . . . . 699, 742, <u>1362</u>
		\KVO@prefix . . . . <u>251</u> , 277, 278, 279, 281, 282, 289, 300, 309, 319, 325, 326, 327, 328, 329, 330, 340, 394, 395, 416, 425, 441, 449
		\KVO@process@pti@ns . 1033, 1050, 1067
		\KVO@process@options . . . . 1009, 1011
		\KVO@ProcessKeyvalOptions . . . . . . . . . . 679, 683, 685
		\KVO@ProcessLocalKeyvalOptions . . . . . . 802, 806, 808
		\KVO@removeelement . . . . . 1335
		\KVO@removegarbage . . . . 1325, 1334
		\KVO@removelastspace . . . . . . . . . 1304, 1312, 1315, 1317, 1321
		\KVO@result 1262, 1264, 1287, 1288, 1294, 1295, 1323, 1324, 1328, 1329

\KVO@SanitizedCurrentOption . . . .	1069, 1092, 1235, 1238, 1258	\OptionNotUsed . . . . .	1094
\KVO@setcurrents . . . . .	865, 871	<b>P</b>	
\KVO@setcurrentvalue . . . . .	878, 881	\PackageError . . . . .	310, 509, 559, 593, 813, 954
\KVO@setkeys . . . . .	266, 792, 849	\PackageInfo . . . . .	123, 376, 409, 474, 530, 546, 581
\KVO@splitcomma . . . . .	1263, 1266, 1270	\PackageWarning . . . . .	283, 363, 464, 604, 623
\KVO@temp . . . . .	634, 640, 687, 700, 702, 712, 713, 729, 736, 769, 773, 790, 794, 810, 820, 827, 831, 847, 851, 971, 978, 984, 988, 994, 1126, 1128, 1141, 1149, 1158	\PackageWarningNoLine . . . . .	57, 946, 962
\KVO@true . . . . .	354, 382	\PassOptionsToPackage . . . . .	63, 80
\KVO@use@option . . . . .	1028, 1044, 1070, 1073, 1082	\ProcessKeyvalOptions . . . . .	4, 74, 675, 886, 1566
\KVO@VOID@ . . . . .	432, 453, 459	\ProcessLocalKeyvalOptions . . . . .	4, 798, 814
\KVO@voidkey . . . . .	433, 454	\ProcessOptions . . . . .	239, 998
\KVO@x . . . . .	1282, 1283, 1288	\protect . . . . .	223
\KVO@xprocess@options . . . . .	1009, 1035	\ProvidesFile . . . . .	1570
\KVOdyn@action . . . . .	508, 511, 515	\ProvidesPackage . . . . .	5, 116, 164, 942, 1552
\KVOdyn@ext . . . . .	495, 498, 502, 567, 578	<b>R</b>	
\KVOdyn@name . . . . .	494, 497, 501, 558, 567, 583	\RangeCatcodeCheck . . . . .	1421, 1449, 1450, 1451, 1452, 1453, 1454, 1455, 1456, 1457, 1458, 1459, 1460
\kvsetkeys . . . . .	274, 506, 1557	\RangeCatcodeInvalid . . . . .	1413, 1441, 1442, 1443, 1444, 1482, 1491, 1492, 1493, 1494
<b>L</b>		\renewcommand . . . . .	93
\LoadClass . . . . .	1227	\repeat . . . . .	1388, 1400, 1411, 1419, 1434, 1480, 1488
\LoadCommand . . . . .	1438, 1448	\RequirePackage . . . . .	7, 8, 84, 217, 219, 220, 236, 951, 956, 1508, 1509, 1515, 1553, 1571
\loop . . . . .	1388, 1404, 1415, 1423, 1473, 1484	\reserved@a . . . . .	1130, 1135, 1155, 1156, 1159, 1161, 1171, 1231
\ltx@empty . . . . .	661, 664	\reserved@b . . . . .	1144, 1152, 1156
\ltx@ifUndefined . . . . .	267	\RestoreCatcodes . . . . .	1402, 1405, 1406, 1461, 1471, 1474, 1475, 1506
\ltx@ifundefined . . . . .	256	<b>S</b>	
\ltx@onelevel@sanitize . . . . .	353, 384, 385, 710, 743	\setkeys . . . . .	44, 268, 1529, 1543, 1561
\ltx@undefined . . . . .	649	\SetupDriver . . . . .	32, 33, 34, 35, 38, 40
<b>M</b>		\SetupKeyvalOptions . . . . .	5, 13, 273, 487, 1534, 1554
\makeatletter . . . . .	1170, 1469, 1514	\space . . . . .	315, 814, 1186, 1189, 1192, 1194, 1198, 1221, 1236, 1239, 1241, 1242, 1427, 1428, 1436
\MCS@driver . . . . .	84	\strip@prefix . . . . .	511, 1180, 1185, 1259
\MCS@emph . . . . .	88, 94	<b>T</b>	
\meaning . . . . .	511, 1181, 1185, 1259	\t . . . . .	1343, 1344
\MessageBreak . . . . .	58, 284, 285, 311, 316, 318, 320, 321, 366, 467, 511, 531, 547, 560, 584, 596, 607, 624, 955, 963, 964, 1188, 1189, 1191, 1192, 1193, 1195, 1197, 1220, 1221, 1222, 1223, 1239	\Test . . . . .	1440, 1463, 1490, 1508, 1509
\msg . . . . .	1516, 1518, 1521, 1524, 1527, 1538, 1541	\TestExpected . . . . .	1574, 1575, 1579
<b>N</b>		\textcolor . . . . .	94
\NeedsTeXFormat . . . . .	4, 893, 1468, 1513, 1551, 1569	\the . . . . .	174, 175, 176, 177, 178, 179, 180, 181, 194, 403, 412, 450, 640, 642, 735, 757, 762, 769, 772, 779, 783, 791, 792, 827, 830, 837, 841, 848, 849, 902, 903, 904, 905, 906, 907, 908, 909, 922, 1024,
\newcommand . . . . .	40, 43, 273, 276, 308, 386, 422, 478, 481, 504, 611, 675, 798	<b>O</b>	
\next . . . . .	424, 426, 429, 1394, 1396, 1398	\on@line . . . . .	1219
\number . . . . .	1429		
\numexpr . . . . .	1113, 1119		

1113, 1119, 1158, 1343, 1346,		<b>W</b>
1348, 1353, 1407, 1427, 1428, 1476	<code>\write</code> .....	120, 149, 1516
<code>\TMP@EnsureCode</code> ... 191, 198, 199,		
200, 201, 202, 203, 204, 205,		<b>X</b>
206, 207, 208, 209, 210, 211,	<code>\x</code> ....	111, 112, 115, 119, 123, 125,
212, 213, 214, 919, 926, 927,		148, 153, 163, 172, 184, 292,
928, 929, 930, 931, 932, 933,		303, 332, 343, 402, 419, 431,
934, 935, 936, 937, 938, 939, 940		444, 457, 459, 507, 517, 520,
<code>\toks</code> ..... 727, 756, 757, 778, 779,		536, 564, 614, 617, 641, 644,
791, 818, 836, 837, 848, 1022, 1024		678, 681, 801, 804, 858, 860,
<code>\toks@</code> ..... 398,		900, 912, 972, 981, 1023, 1026,
400, 403, 405, 412, 446, 450,		1055, 1056, 1060, 1065, 1084,
639, 640, 642, 729, 731, 735,		1091, 1177, 1179, 1189, 1195,
736, 761, 762, 768, 769, 772,		1338, 1339, 1341, 1348, 1352, 1355
773, 782, 783, 792, 820, 822,	<code>\XKV@classoptionslist</code> .	663, 664, 666
826, 827, 830, 831, 840, 841,	<code>\XKV@documentclass</code> .....	660, 661
849, 1017, 1020, 1024, 1157,		
1158, 1337, 1343, 1346, 1348, 1353		<b>Y</b>
<code>\tw@</code> ..... 727, 756, 757, 778, 779,	<code>\y</code> .....	1184, 1185, 1192, 1195
791, 818, 836, 837, 848, 1022, 1024		
<code>\typeout</code> ..... 1576, 1578, 1579		
		<b>Z</b>
<b>U</b>		
<code>\unexpanded</code> ..... 955	<code>\zap@space</code> ....	877, 1155, 1282, 1296