

# The kvoptions package

Heiko Oberdiek  
<heiko.oberdiek at gmail.com>

2010/02/22 v3.7

## 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>3</b>
2.1	Process options . . . . .	3
2.1.1	<code>\ProcessKeyvalOptions</code> . . . . .	3
2.1.2	<code>\ProcessLocalKeyvalOptions</code> . . . . .	4
2.1.3	<code>\SetupKeyvalOptions</code> . . . . .	4
2.2	Option declarations . . . . .	4
2.2.1	<code>\DeclareStringOption</code> . . . . .	4
2.2.2	<code>\DeclareBoolOption</code> . . . . .	5
2.2.3	<code>\DeclareComplementaryOption</code> . . . . .	6
2.2.4	<code>\DeclareVoidOption</code> . . . . .	6
2.2.5	<code>\DeclareDefaultOption</code> . . . . .	6
2.2.6	Local options . . . . .	7
2.2.7	Dynamic options . . . . .	7
2.2.8	<code>\DisableKeyvalOption</code> . . . . .	7
2.2.9	<code>\AddToKeyvalOption</code> . . . . .	8
2.3	Global vs. local options . . . . .	8
2.4	Summary of internal macros . . . . .	9
2.5	plain- <code>T<sub>E</sub>X</code> . . . . .	9
<b>3</b>	<b>Example</b>	<b>9</b>
<b>4</b>	<b>Package options</b>	<b>11</b>
4.1	Package kvoptions-patch . . . . .	11
4.2	Option debugshow . . . . .	12
<b>5</b>	<b>Limitations</b>	<b>12</b>
5.1	Compatibility . . . . .	12
5.1.1	Package kvoptions-patch vs. package xkvltxp . . . . .	12
5.2	Limitations . . . . .	13
5.2.1	Option comparisons . . . . .	13
5.2.2	Option list parsing with package kvoptions-patch . . . . .	13

<b>6</b>	<b>Implementation</b>	<b>13</b>
6.1	Preamble	13
6.2	Option declaration macros	16
6.2.1	<code>\SetupKeyvalOptions</code>	16
6.2.2	<code>\DeclareBoolOption</code>	17
6.2.3	<code>\DeclareStringOption</code>	19
6.2.4	<code>\DeclareVoidOption</code>	20
6.2.5	<code>\DeclareDefaultOption</code>	21
6.2.6	<code>\DeclareLocalOptions</code>	21
6.3	Dynamic options	21
6.3.1	<code>\DisableKeyvalOption</code>	21
6.4	Change option code	23
6.4.1	<code>\AddToKeyvalOption</code>	23
6.5	Process options	24
6.5.1	<code>\ProcessKeyvalOptions</code>	24
6.5.2	<code>\ProcessLocalKeyvalOptions</code>	26
6.5.3	Helper macros	27
6.6	plain- <code>T<sub>E</sub>X</code>	28
6.7	Package <code>kvoptions-patch</code>	28
<b>7</b>	<b>Test</b>	<b>36</b>
7.1	Preface for standard catcode check	36
7.2	Catcode checks for loading	36
<b>8</b>	<b>Installation</b>	<b>39</b>
8.1	Download	39
8.2	Bundle installation	39
8.3	Package installation	39
8.4	Refresh file name databases	40
8.5	Some details for the interested	40
<b>9</b>	<b>References</b>	<b>40</b>
<b>10</b>	<b>History</b>	<b>41</b>
	[0000/00/00 v0.0]	41
	[2004/02/22 v1.0]	41
	[2006/02/16 v2.0]	41
	[2006/02/20 v2.1]	41
	[2006/06/01 v2.2]	41
	[2006/08/17 v2.3]	41
	[2006/08/22 v2.4]	42
	[2007/04/11 v2.5]	42
	[2007/05/06 v2.6]	42
	[2007/06/11 v2.7]	42
	[2007/10/02 v2.8]	42
	[2007/10/11 v2.9]	42
	[2007/10/18 v3.0]	42
	[2009/04/10 v3.1]	42
	[2009/07/17 v3.2]	42
	[2009/07/21 v3.3]	42
	[2009/08/13 v3.4]	42
	[2009/12/04 v3.5]	42
	[2009/12/08 v3.6]	42
	[2010/02/22 v3.7]	43
<b>11</b>	<b>Index</b>	<b>43</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 and used in other packages, e.g. package `helvet` that uses this interface for its option `scaled`.

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

## 1.2 Overview

Package `kvoptions` connects package `keyval` with  $\text{\LaTeX}$ ’s package and class `options`:

Package <code>keyval</code>	Package <code>kvoptions</code>	$\text{\LaTeX}$ kernel
<code>\define@key</code>	<code>\DeclareVoidOption</code> <code>\DeclareStringOption</code> <code>\DeclareBoolOption</code> <code>\DeclareComplementaryOption</code> <code>\DisableKeyvalOption</code>	<code>\DeclareOption</code>
	<code>\DeclareDefaultOption</code>	<code>\DeclareOption*</code>
	<code>\ProcessKeyvalOptions</code>	<code>\ProcessOptions*</code>
	Option patch	Class/package option system
	<code>\SetupKeyvalOptions</code>	

# 2 Usage

## 2.1 Process options

### 2.1.1 `\ProcessKeyvalOptions`

<code>\ProcessKeyvalOptions {<i>family</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  $\text{\LaTeX}$ ’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`

```
\ProcessLocalKeyvalOptions {<family>}
\ProcessLocalKeyvalOptions *
```

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>
}
```

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

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

## 2.2 Option declarations

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

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

### 2.2.1 `\DeclareStringOption`

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

A macro is created that remembers the value of the key `<key>`. The name of the macro consists of the option name `<key>` that is prefixed by the prefix (see 2.1.3). 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 to different switches, they should share one. `\DeclareComplementaryOption` allows this. The option `<key>` shares the switch of option `<parent>`. Example:

```
\DeclareBoolOption{draft}  
\DeclareComplementaryOption{final}{draft}
```

Then `final` sets the switch of `draft` to `false`, and `final=false` enables the `draft` switch.

### 2.2.4 `\DeclareVoidOption`

```
\DeclareVoidOption{<key>}{<code>}
```

`\ProcessKeyvalOptions` can be extended to recognize options that are declared in traditional way by `\DeclareOption`. But in case of the error that the user specifies a value, then this option would not be recognized as key value option because of `\DeclareOption` and not detected as traditional option because of the value part. The user would get an unknown option error, difficult to understand.

`\DeclareVoidOption` solves this problem. It defines the option `<key>` as key value option. If the user specifies a value, a warning is given and the value is ignored.

The code part `<code>` is stored in a macro. The name of the macro consists of the option name `<key>` that is prefixed by the prefix (see 2.1.3). 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

<pre>\DeclareLocalOption {⟨option⟩} \DeclareLocalOptions {⟨option list⟩}</pre>
--

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`

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

`\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}%
}

\ifffoobar@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

<pre> \AddToKeyvalOption {&lt;family&gt;} {&lt;key&gt;} {&lt;code&gt;} \AddToKeyvalOption * {&lt;key&gt;} {&lt;code&gt;} </pre>
---

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

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

## 2.5 plain-TeX

Package `keyval` is also usable in plain-TeX with the help of file `miniltx.tex`. Some features of this package `kvoptions` might also be useful for plain-TeX. If `LATEX` is not found, `\ProcessKeyvalOptions` and option `patch` are disabled. Before using the option declaration commands `\Declare...Option`, `\SetupKeyvalOptions` must be used.

## 3 Example

The following example defined a package that serves some private color management. A boolean option `print` enables print mode without colors. An option `emph` redefines `\emph` to print in the given color. And the driver can be specified by option `driver`.

```
1 *example
2   % Package identification
3   % -----
4   \NeedsTeXFormat{LaTeX2e}
5   \ProvidesPackage{example-mycolorsetup}[2010/02/22 Managing my colors]
6
7   \RequirePackage{ifpdf}
8   \RequirePackage{kvoptions}
9
10  % Option declarations
11  % -----
12
13  \SetupKeyvalOptions{
14    family=MCS,
15    prefix=MCS@
16  }
17  % Use a shorter family name and prefix
18
19  % Option print
20  \DeclareBoolOption{print}
21  % is the same as
22  % \DeclareBoolOption[false]{print}
23
24  % Option driver
25  \ifpdf
26    \DeclareStringOption[pdftex]{driver}
27  \else
28    \DeclareStringOption[dvips]{driver}
29  \fi
```

```

30
31 % Alternative interface for driver options
32 \DeclareVoidOption{dvips}{\SetupDriver}
33 \DeclareVoidOption{dvipdfm}{\SetupDriver}
34 \DeclareVoidOption{pdftex}{\SetupDriver}
35 % In \SetupDriver we take the current option \CurrentOption
36 % and pass it to the driver option.
37 % The \expandafter commands expand \CurrentOption at the
38 % time, when \SetupDriver is executed and \CurrentOption
39 % has the correct meaning.
40 \newcommand*{\SetupDriver}{%
41 \expandafter\@SetupDriver\expandafter{\CurrentOption}%
42 }
43 \newcommand*{\@SetupDriver}[1]{%
44 \setkeys{MCS}{driver={#1}}%
45 }
46
47 % Option emph
48 % An empty value means, we want to have no color for \emph.
49 % If the user specifies option emph without value, the red is used.
50 \DeclareStringOption{emph}[red]
51 % is the same as
52 % \DeclareStringOption[] {emph} [red]
53
54 % Default option rule
55 \DeclareDefaultOption{%
56 \ifx\CurrentOptionValue\relax
57 \PackageWarningNoLine{\@currname}{%
58 Unknown option '\CurrentOption'\MessageBreak
59 is passed to package 'color'%
60 }%
61 % Pass the option to package color.
62 % Again it is better to expand \CurrentOption.
63 \expandafter\PassOptionsToPackage
64 \expandafter{\CurrentOption}{color}%
65 \else
66 % Package color does not take options with values.
67 % We provide the standard LaTeX error.
68 \@unknownoptionerror
69 \fi
70 }
71
72 % Process options
73 % -----
74 \ProcessKeyvalOptions*
75
76 % Implementation depending on option values
77 % -----
78 % Code for print mode
79 \ifMCS@print
80 \PassOptionsToPackage{monochrome}{color}
81 % tells package color to use black and white
82 \fi
83
84 \RequirePackage[\MCS@driver]{color}
85 % load package color with the correct driver
86
87 % \emph setup
88 \ifx\MCS@emph\@empty
89 % \@empty is a predefined macro with empty contents.
90 % the option value of option emph is empty, thus
91 % we do not want a redefinition of \emph.

```

```

92 \else
93   \renewcommand*{\emph}[1]{%
94     \textcolor{\MCS@emph}{#1}%
95   }
96 \fi
97 \example

```

## 4 Package options

The package `kvoptions` knows two package options `patch` and `debugshow`. The options of package `kvoptions` are intended for authors, not for package/class writers. Inside a package it is too late for option `patch` and `debugshow` enables some messages that are perhaps useful for the debugging phase. Also  $\LaTeX$  is unhappy if a package is loaded later again with options that are previously not given. Thus package and class authors, stay with `\RequirePackage{kvoptions}` without options.

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

### 4.1 Package `kvoptions-patch`

$\LaTeX$ 's system of package/class options has some severe limitations that especially affects the value part if options are used as pair of key and value.

- Spaces are removed, regardless where:

```
\documentclass[box=0 0 400 600]{article}
```

Now each package will see `box=00400600` as global option.

- In the previous case also braces would not help:

```
\documentclass[box={0 0 400 600}]{article}
```

The result is an error message:

```
! LaTeX Error: Missing \begin{document}.
```

As local option, however, it works if the package knows about key value options (By using this package, for example).

- The requirements on robustness are extremely high.  $\LaTeX$  expands the option. All that will not work as environment name will break also as option. Even a `\relax` will generate an error message:

```
! Missing \endcsname inserted.
```

Of course,  $\LaTeX$  does not use its protecting mechanisms. On contrary `\protect` itself will cause errors.

- The options are expanded. But perhaps the package will do that, because it has to setup some things before? Example `hyperref`:

```
\usepackage[pdfauthor=M"uller]{hyperref}
```

Package `hyperref` does not see `M"uller` but its expansion and it does not like it, you get many warnings

```
Token not allowed in a PDFDocEncoded string
```

And the title becomes: `Mu127uller`. Therefore such options must usually be given after package `hyperref` is loaded:

```
\usepackage{hyperref}
\hypersetup{pdfauthor=Fran\c coise M"uller}
```

As package option it will even break with `Fran\c coise` because of the cedilla `\c c`, it is not robust enough.

For users that do not want with this limitations the package offers option `patch`. It patches  $\LaTeX$ 's option system and tries to teach it also to handle options that are given as pairs of key and value and to prevent expansion. It can already be used at the very beginning, before `\documentclass`:

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

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

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

Some remarks:

- The patch requires  $\epsilon$ - $\TeX$ , its `\unexpanded` feature is much to nice. It is possible to work around using token registers. But the code becomes longer, slower, more difficult to read and maintain. The package without option `patch` works and will work without  $\epsilon$ - $\TeX$ .
- The code for the patch is quite long, there are many test cases. Thus the probability for bugs is probably not too small.

## 4.2 Option `debugshow`

The name of this option follows the convention of packages `multicol`, `tabularx`, and `tracefmt`. Currently it prints the setting of boolean options, declared by `\DeclareBoolOption` in the `.log` file, if that boolean option is used. You can activate the option by

- `\PassOptionsToPackage{debugshow}{kvoptions}`  
Put this somewhere before package `kvoptions` is loaded first, e.g. before `\documentclass`.
- `\RequirePackage[debugshow]{kvoptions}`  
Before `\documentclass` even an author has to use `\RequirePackage`. `\usepackage` only works after `\documentclass`.

The preferred method is `\PassOptionsToPackage`, because it does not force the package loading and does not disturb, if the package is not loaded later at all.

# 5 Limitations

## 5.1 Compatibility

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

Package `xkvltxp` from the `xkeyval` project has the same goal as package `kvoptions-patch` and to patch  $\LaTeX$ '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  $\LaTeX$  compares option lists, e.g. option clash check, `\@ifpackagewith`, or `\@ifclasswith`. Apart from catcode and sanitizing problems of option `patch`, there is another problem.  $\LaTeX$  does not know about the type and default values of options in key value style. Thus an option clash is reported, even if the key value has the same meaning:

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

### 5.2.2 Option list parsing with 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  $\LaTeX$ .

```
99 \begingroup
```

```

100 \catcode44 12 % ,
101 \catcode45 12 % -
102 \catcode46 12 % .
103 \catcode58 12 % :
104 \catcode64 11 % @
105 \catcode123 1 % {
106 \catcode125 2 % }
107 \expandafter\let\expandafter\x\csname ver@kvoptions.sty\endcsname
108 \ifx\x\relax % plain-TeX, first loading
109 \else
110   \def\empty{}%
111   \ifx\x\empty % LaTeX, first loading,
112     % variable is initialized, but \ProvidesPackage not yet seen
113   \else
114     \catcode35 6 % #
115     \expandafter\ifx\csname PackageInfo\endcsname\relax
116       \def\x#1#2{%
117         \immediate\write-1{Package #1 Info: #2.}%
118       }%
119     \else
120       \def\x#1#2{\PackageInfo{#1}{#2, stopped}}%
121     \fi
122     \x{kvoptions}{The package is already loaded}%
123     \aftergroup\endinput
124   \fi
125 \fi
126 \endgroup

```

Package identification:

```

127 \begingroup
128 \catcode35 6 % #
129 \catcode40 12 % (
130 \catcode41 12 % )
131 \catcode44 12 % ,
132 \catcode45 12 % -
133 \catcode46 12 % .
134 \catcode47 12 % /
135 \catcode58 12 % :
136 \catcode64 11 % @
137 \catcode91 12 % [
138 \catcode93 12 % ]
139 \catcode123 1 % {
140 \catcode125 2 % }
141 \expandafter\ifx\csname ProvidesPackage\endcsname\relax
142   \def\x#1#2#3[#4]{\endgroup
143     \immediate\write-1{Package: #3 #4}%
144     \xdef#1{#4}%
145   }%
146 \else
147   \def\x#1#2[#3]{\endgroup
148     #2[#{#3}]%
149     \ifx#1\@undefined
150       \xdef#1{#3}%
151     \fi
152     \ifx#1\relax
153       \xdef#1{#3}%
154     \fi
155   }%
156 \fi
157 \expandafter\x\csname ver@kvoptions.sty\endcsname
158 \ProvidesPackage{kvoptions}%
159 [2010/02/22 v3.7 Keyval support for LaTeX options (H0)]

```

## Catcodes

```
160 \begingroup
161 \catcode123 1 % {
162 \catcode125 2 % }
163 \def\x{\endgroup
164 \expandafter\edef\csname KVO@AtEnd\endcsname{%
165 \catcode35 \the\catcode35\relax
166 \catcode64 \the\catcode64\relax
167 \catcode123 \the\catcode123\relax
168 \catcode125 \the\catcode125\relax
169 }%
170 }%
171 \x
172 \catcode35 6 % #
173 \catcode64 11 % @
174 \catcode123 1 % {
175 \catcode125 2 % }
176 \def\TMP@EnsureCode#1#2{%
177 \edef\KVO@AtEnd{%
178 \KVO@AtEnd
179 \catcode#1 \the\catcode#1\relax
180 }%
181 \catcode#1 #2\relax
182 }
183 \TMP@EnsureCode{1}{14}% ^^A (comment)
184 \TMP@EnsureCode{2}{14}% ^^A (comment)
185 \TMP@EnsureCode{33}{12}% !
186 \TMP@EnsureCode{39}{12}% '
187 \TMP@EnsureCode{40}{12}% (
188 \TMP@EnsureCode{41}{12}% )
189 \TMP@EnsureCode{42}{12}% *
190 \TMP@EnsureCode{44}{12}% ,
191 \TMP@EnsureCode{45}{12}% -
192 \TMP@EnsureCode{46}{12}% .
193 \TMP@EnsureCode{47}{12}% /
194 \TMP@EnsureCode{58}{12}% :
195 \TMP@EnsureCode{61}{12}% =
196 \TMP@EnsureCode{62}{12}% >
197 \TMP@EnsureCode{94}{7}% ^ (superscript)
198 \TMP@EnsureCode{96}{12}% ‘
```

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

```
199 \@ifundefined{define@key}{%
200 \RequirePackage{keyval}\relax
201 }{}
```

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

```
202 \RequirePackage{kvsetkeys}[2007/09/29]
```

## Provide macros for plain-TeX.

```
203 \@ifundefined{@onelevel@sanitize}{%
204 \def{@onelevel@sanitize#1}{%
205 \edef#1{\expandafter\strip@prefix\meaning#1}%
206 }%
207 }{ }
208 \@ifundefined{strip@prefix}{%
209 \def\strip@prefix#1>{}}%
```

```

210 }{}
211 \@ifundefined{@x@protect}{%
212   \def\x@protect#1\fi#2#3{%
213     \fi\protect#1%
214   }%
215   \let\@typeset@protect\relax
216 }{}
217 \@ifundefined{@currname}{%
218   \def\@currname{}%
219 }{}
220 \@ifundefined{@current}{%
221   \def\@current{}%
222 }{}

```

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

```

223 \DeclareOption{debugshow}{\catcode\@ne=9 }
224 \DeclareOption{patch}{%
225   \AtEndOfPackage{%
226     \RequirePackage{kvoptions-patch}[2010/02/22]%
227   }%
228 }

```

Optionen auswerten:

```

229 \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.

`\KVO@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.

```

230 \define@key{KVO}{family}{%
231   \expandafter\edef\csname KVO@family@%
232     \@currname.\@current\endcsname{#1}%
233 }
234 \def\KVO@family{%
235   \@ifundefined{KVO@family@\@currname.\@current}{%
236     \@currname
237   }{%
238     \csname KVO@family@\@currname.\@current\endcsname
239   }%
240 }

```

`\KVO@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.

```

241 \define@key{KVO}{prefix}{%
242   \expandafter\edef\csname KVO@prefix@%
243     \@currname.\@current\endcsname{#1}%
244 }
245 \def\KVO@prefix{%
246   \@ifundefined{KVO@prefix@\@currname.\@current}{%

```



```

247   \@currname @%
248   }{%
249   \csname KVO@prefix@\@currname.\@current\endcsname
250   }%
251   }

```

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

```

252 \newcommand*\SetupKeyvalOptions}{%
253   \setkeys{KVO}%
254 }

```

### 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.

```

255 \newcommand*\DeclareBoolOption}[2][false]{%
256   \KVO@ifdefinable{if\KVO@prefix#2}{%
257     \KVO@ifdefinable{\KVO@prefix#2true}{%
258       \KVO@ifdefinable{\KVO@prefix#2false}{%
259         \csname newif\expandafter\endcsname
260         \csname if\KVO@prefix#2\endcsname
261         \@ifundefined{\KVO@prefix#2#1}{%
262           \PackageWarning{kvoptions}{%
263             Initialization of option ‘#2’ failed,\MessageBreak
264             cannot set boolean option to ‘#1’,\MessageBreak
265             use ‘true’ or ‘false’, now using ‘false’%
266           }%
267         }{%
268           \csname\KVO@prefix#2#1\endcsname
269         }%
270         \begingroup
271         \edef\x{\endgroup
272           \noexpand\define@key{\KVO@family}{#2}[true]{%
273             \noexpand\KVO@boolkey{\@currname}%
274             \ifx\@current\@clsextension
275               \noexpand\@clsextension
276             \else
277               \noexpand\@pkgextension
278             \fi
279             {\KVO@prefix}{#2}{###1}%
280           }%
281         }%
282         \x
283       }%
284     }%
285   }%
286 }

```

`\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.

```

287 \newcommand*\DeclareComplementaryOption}[2]{%
288   \@ifundefined{if\KVO@prefix#2}{%

```

```

289 \PackageError{kvoptions}{%
290   Cannot generate option code for ‘#1’,\MessageBreak
291   parent switch ‘#2’ does not exist%
292 }{%
293   You are inside %
294   \ifx\@currentx\@clsextension class\else package\fi\space
295   ‘\@currname.\@currentx’.\MessageBreak
296   ‘\KVO@family’ is used as family %
297   for the keyval options.\MessageBreak
298   ‘\KVO@prefix’ serves as prefix %
299   for internal switch macros.\MessageBreak
300   \MessageBreak
301   \@ehc
302 }%
303 }{%
304   \KVO@ifdefinable{\KVO@prefix#1true}{%
305     \KVO@ifdefinable{\KVO@prefix#1false}{%
306       \expandafter\let\csname\KVO@prefix#1false\expandafter\endcsname
307       \csname\KVO@prefix#2true\endcsname
308     \expandafter\let\csname\KVO@prefix#1true\expandafter\endcsname
309     \csname\KVO@prefix#2false\endcsname

```

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

```

310   \begingroup
311   \edef\x{\endgroup
312     \noexpand\define@key{\KVO@family}{#1}[true]{%
313       \noexpand\KVO@boolkey{\@currname}%
314       \ifx\@currentx\@clsextension
315         \noexpand\@clsextension
316       \else
317         \noexpand\@pkgextension
318       \fi
319       {\KVO@prefix}{#1}{###1}%
320     }%
321   }%
322   \x
323 }%
324 }%
325 }%
326 }

```

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

```

327 \def\KVO@ifdefinable#1{%
328   \expandafter\@ifdefinable\csname #1\endcsname
329 }

```

`\KVO@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

```

```

330 \def\KVO@boolkey#1#2#3#4#5{%
331   \edef\KVO@param{#5}%
332   \@onelevel@sanitize\KVO@param
333   \ifx\KVO@param\KVO@true
334     \expandafter\@firstofone
335   \else
336     \ifx\KVO@param\KVO@false
337       \expandafter\expandafter\expandafter\@firstofone

```

```

338 \else
339 \ifx#2\@clsextension
340 \expandafter\ClassWarning
341 \else
342 \expandafter\PackageWarning
343 \fi
344 {#1}{%
345 Value '\KVO@param' is not supported by\MessageBreak
346 option '#4'%
347 }%
348 \expandafter\expandafter\expandafter\@gobble
349 \fi
350 \fi
351 {%
352 ^^A\ifx#2\@clsextension
353 ^^A \expandafter\ClassInfo
354 ^^A\else
355 ^^A \expandafter\PackageInfo
356 ^^A\fi
357 ^^A{#1}{{[option] #4=\KVO@param}%
358 \csname#3#4\KVO@param\endcsname
359 }%
360 }

```

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

```

361 \def\KVO>true{true}
362 \def\KVO>false{false}
363 \@onelevel@sanitize\KVO>true
364 \@onelevel@sanitize\KVO>false

```

### 6.2.3 \DeclareStringOption

\DeclareStringOption

```

365 \newcommand*{\DeclareStringOption}[2][ ]{%
366 \@ifnextchar[{%
367 \KVO@DeclareStringOption{#1}{#2}@%
368 }{%
369 \KVO@DeclareStringOption{#1}{#2}{ }[%
370 ]%
371 }

```

\KVO@DeclareStringOption

```

372 \def\KVO@DeclareStringOption#1#2#3[#4]{%
373 \KVO@ifdefinable{\KVO@prefix#2}{%
374 \@namedef{\KVO@prefix#2}{#1}%
375 \begingroup
376 \ifx\#3\%
377 \toks@{ }%
378 \else
379 \toks@{[#4]}%
380 \fi
381 \edef\x{\endgroup
382 \noexpand\define@key{\KVO@family}{#2}\the\toks@{
383 ^^A\begingroup
384 ^^A \toks@{####1}%
385 ^^A \ifx\@current\@clsextension
386 ^^A \noexpand\ClassInfo
387 ^^A \else
388 ^^A \noexpand\PackageInfo
389 ^^A \fi
390 ^^A {\@currname}{%

```

```

391      ^^A [option] #2={\noexpand\the\toks@}%
392      ^^A }%
393      ^^A\endgroup
394      \noexpand\def
395      \expandafter\noexpand\cname\KVO@prefix#2\endcsname{###1}%
396      }%
397      }%
398      \x
399      }%
400 }

```

## 6.2.4 \DeclareVoidOption

\DeclareVoidOption

```

401 \newcommand*{\DeclareVoidOption}[1]{%
402   \begingroup
403   \let\next@gobbletwo
404   \KVO@ifdefinable{\KVO@prefix#1}{%
405     \let\next@firstofone
406   }%
407   \expandafter\endgroup
408   \next{%
409     \begingroup
410     \edef\x{\endgroup
411       \noexpand\define@key{\KVO@family}{#1}{\KVO@VOID@}{%
412         \noexpand\KVO@voidkey{\@currname}%
413         \ifx\@current\@clsextension
414           \noexpand\@clsextension
415         \else
416           \noexpand\@pkgextension
417         \fi
418         {#1}%
419         {###1}%
420         \expandafter\noexpand\cname\KVO@prefix#1\endcsname
421       }%
422     }%
423     \x
424     \@namedef{\KVO@prefix#1}%
425   }%
426 }
427 \def\KVO@VOID@{@VOID@}

```

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

```

428 \def\KVO@voidkey#1#2#3#4{%
429   \def\CurrentOption{#3}%
430   \begingroup
431   \def\x{#4}%
432   \expandafter\endgroup
433   \ifx\x\KVO@VOID@
434   \else
435     \if#2\@clsextension
436       \expandafter\ClassWarning
437     \else
438       \expandafter\PackageWarning
439     \fi
440     {#1}{%
441       Unexpected value for option ‘#3’\MessageBreak
442       is ignored%

```

```

443   }%
444 \fi
445 ^^A\ifx#2\@clsextension
446 ^^A \expandafter\ClassInfo
447 ^^A\else
448 ^^A \expandafter\PackageInfo
449 ^^A\fi
450 ^^A{#1}{[option] #3}%
451 }

```

### 6.2.5 \DeclareDefaultOption

\DeclareDefaultOption

```

452 \newcommand*\DeclareDefaultOption{%
453 \namedef{KV0@default@\currname.\@currentx}%
454 }

```

### 6.2.6 \DeclareLocalOptions

\DeclareLocalOptions

```

455 \newcommand*\DeclareLocalOptions}[1]{%
456 \comma@parse{#1}\KV0@DeclareLocalOption
457 }

```

\KV0@DeclareLocalOption

```

458 \def\KV0@DeclareLocalOption#1{%
459 \expandafter\def\csname KV0@local@\KV0@family @#1\endcsname}%
460 }

```

## 6.3 Dynamic options

### 6.3.1 \DisableKeyvalOption

```

461 \SetupKeyvalOptions{%
462 family=KV0dyn,%
463 prefix=KV0dyn@%
464 }
465 \DeclareBoolOption[true]{global}
466 \DeclareComplementaryOption{local}{global}
467 \DeclareStringOption[undef]{action}
468 \let\KV0dyn@name\relax
469 \let\KV0dyn@ext\@empty
470 \define@key{KV0dyn}{class}{%
471 \def\KV0dyn@name{#1}%
472 \let\KV0dyn@ext\@clsextension
473 }
474 \define@key{KV0dyn}{package}{%
475 \def\KV0dyn@name{#1}%
476 \let\KV0dyn@ext\@pkgextension
477 }
478 \newcommand*\DisableKeyvalOption}[3][ ]{%
479 \begingroup
480 \setkeys{KV0dyn}{#1}%
481 \def\x{\endgroup}%
482 \ifundefined{KV0@action@\KV0dyn@action}{%
483 \PackageError{kvoptions}{%
484 Unknown disable action %
485 '\expandafter\strip@prefix\meaning\KV0dyn@action'\MessageBreak
486 for option '#3' in keyval family '#2'%
487 } \@ehc
488 }{%
489 \csname KV0@action@\KV0dyn@action\endcsname{#2}{#3}%

```

```

490   }%
491   \x
492 }
493 \def\KVO@action@undef#1#2{%
494   \edef\x{\endgroup
495     \ifKV0dyn@global\global\fi
496     \let
497     \expandafter\noexpand\csname KV@#1@#2\endcsname
498     \relax
499     \ifKV0dyn@global\global\fi
500     \let
501     \expandafter\noexpand\csname KV@#1@#2@default\endcsname
502     \relax
503   }%
504   ^^A\PackageInfo{kvoptions}{%
505     ^^A [option] key ‘#2’ of family ‘#1’\MessageBreak
506     ^^A is disabled (undef, \ifKV0dyn@global\global\else local\fi)%
507   ^^A}%
508 }
509 \def\KVO@action@ignore#1#2{%
510   \edef\x{\endgroup
511     \ifKV0dyn@global\global\fi
512     \let
513     \expandafter\noexpand\csname KV@#1@#2\endcsname
514     \noexpand@gobble
515     \ifKV0dyn@global\global\fi
516     \let
517     \expandafter\noexpand\csname KV@#1@#2@default\endcsname
518     \noexpand@empty
519   }%
520   ^^A\PackageInfo{kvoptions}{%
521     ^^A [option] key ‘#2’ of family ‘#1’\MessageBreak
522     ^^A is disabled (ignore, \ifKV0dyn@global\global\else local\fi)%
523   ^^A}%
524 }
525 \def\KVO@action@error{%
526   \KVO@do@action{error}%
527 }
528 \def\KVO@action@warning{%
529   \KVO@do@action{warning}%
530 }
#1  error or warning
#2  <family>
#3  <key>
531 \def\KVO@do@action#1#2#3{%
532   \ifx\KV0dyn@name\relax
533     \PackageError{kvoptions}{%
534       Action type ‘#1’ needs package/class name\MessageBreak
535       for key ‘#3’ in family ‘#2’%
536     }\@ehc
537   \else
538     \edef\x{\endgroup
539       \noexpand\define@key{#2}{#3}[] {%
540         \expandafter\noexpand\csname KV0@disable@#1\endcsname
541         {\KV0dyn@name}\noexpand\KV0dyn@ext{#3}%
542       }%
543     \ifKV0dyn@global
544       \global\let
545       \expandafter\noexpand\csname KV@#2@#3\endcsname
546       \expandafter\noexpand\csname KV@#2@#3\endcsname
547       \global\let
548       \expandafter\noexpand\csname KV@#2@#3@default\endcsname

```

```

549     \expandafter\noexpand\csname KV@#2@#3@default\endcsname
550     \fi
551   }%
552   ^^A\ifx\KV0dyn@ext\@clsextension
553   ^^A \expandafter\ClassInfo
554   ^^A\else
555   ^^A \expandafter\PackageInfo
556   ^^A\fi
557   ^^A{\KV0dyn@name}{%
558   ^^A [option] key ‘#3’ of family ‘#2’\MessageBreak
559   ^^A is disabled (#1, \ifKV0dyn@global global\else local\fi)%
560   ^^A}%
561 \fi
562 }
563 \def\KV0@disable@error#1#2#3{%
564 \ifx#2\@clsextension
565 \expandafter\ClassError
566 \else
567 \expandafter\PackageError
568 \fi
569 {#1}{%
570 Option ‘#3’ is given too late,\MessageBreak
571 now the option is ignored%
572 }\@ehc
573 }
574 \def\KV0@disable@warning#1#2#3{%
575 \ifx#2\@clsextension
576 \expandafter\ClassWarning
577 \else
578 \expandafter\PackageWarning
579 \fi
580 {#1}{%
581 Option ‘#3’ is already consumed\MessageBreak
582 and has no effect%
583 }%
584 }

```

## 6.4 Change option code

### 6.4.1 \AddToKeyvalOption

\AddToKeyvalOption

```

585 \newcommand*\AddToKeyvalOption{%
586 \ifstar%
587 \begingroup
588 \edef\x{\endgroup
589 \noexpand\KV0@AddToKeyvalOption{\KV0@family}%
590 }%
591 \x
592 }%
593 \KV0@AddToKeyvalOption
594 }

```

\KV0@AddToKeyvalOption

```

595 \def\KV0@AddToKeyvalOption#1#2{%
596 \ifundefined{KV@#1@#2}{%
597 \PackageWarning{kvoptions}{%
598 Key ‘#2’ of family ‘#1’ does not exist.\MessageBreak
599 Ignoring \string\AddToKeyvalOption
600 }%
601 \@gobble
602 }{%
603 \edef\KV0@next{%

```

```

604     \noexpand\KVO@@AddToKeyvalOption
605     \expandafter\noexpand\csname KV@#1@#2\endcsname
606   }%
607   \afterassignment\KVO@next
608   \def\KVO@temp##1%
609 }%
610 }

```

\KVO@@AddToKeyvalOption

```

611 \def\KVO@@AddToKeyvalOption#1{%
612   \begingroup
613   \toks@\expandafter{#1{##1}}%
614   \toks@\expandafter{\the\expandafter\toks@\KVO@temp{##1}}%
615   \edef\x{\endgroup
616     \noexpand\def\noexpand#1###1{\the\toks@}%
617   }%
618   \x
619 }

```

## 6.5 Process options

### 6.5.1 \ProcessKeyvalOptions

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

```

620 \newcommand*{\ProcessKeyvalOptions}{%
621   \@ifstar{%
622     \begingroup
623     \edef\x{\endgroup
624       \noexpand\KVO@ProcessKeyvalOptions{\KVO@family}%
625     }%
626     \x
627   }%
628   \KVO@ProcessKeyvalOptions
629 }

630 \def\KVO@ProcessKeyvalOptions#1{%
631   \let\@tempc\relax
632   \let\KVO@temp\@empty

```

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

```

633   \ifx\@current\@clsextension
634   \else
635     \ifx\@classoptionslist\relax
636     \else
637       \@for\KVO@CurrentOption:=\@classoptionslist\do{%
638         \ifundefined{KV@#1}\expandafter\KVO@getkey
639           \KVO@CurrentOption=\@nil}{%
640       }{%
641         \ifundefined{KVO@local@#1}\expandafter\KVO@getkey
642           \KVO@CurrentOption=\@nil}{%
643         \ifx\KVO@Patch Y%
644           \edef\KVO@temp{%
645             \etex@unexpanded\expandafter{%
646               \KVO@temp
647             }%
648           ,%
649           \etex@unexpanded\expandafter{%
650             \KVO@CurrentOption
651           }%
652           ,%
653         }%

```



```

654         \@onelevel@sanitize\KVO@CurrentOption
655     \else
656         \edef\KVO@temp{%
657             \KVO@temp
658             ,%
659             \KVO@CurrentOption
660             ,%
661         }%
662     \fi
663     \@expandtwoargs\@removeelement\KVO@CurrentOption
664     \@unusedoptionlist\@unusedoptionlist
665 }{}%
666 }%
667 }%
668 \fi
669 \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`.

```

670 \begingroup
671   \toks\tw@{}%
672   \@ifundefined{opt@\@currname.\@current}{%
673     \toks@\expandafter{\KVO@temp}%
674   }{%
675     \toks@\expandafter\expandafter\expandafter{%
676       \csname opt@\@currname.\@current\endcsname
677     }%
678     \ifx\@current\@clsextension
679       \edef\CurrentOption{\the\toks@}%
680       \toks@\expandafter{\KVO@temp}%
681       \@for\CurrentOption:=\CurrentOption\do{%
682         \@ifundefined{%
683           KV@#1@\expandafter\KVO@getkey\CurrentOption=@nil
684         }{%

```

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

```

685     \@ifundefined{KVO@default@\@currname.\@current}{%
686       \ifx\KVO@Patch Y%
687         \@onelevel@sanitize\CurrentOption
688       \fi
689       \ifx\@unusedoptionlist\@empty
690         \global\let\@unusedoptionlist\CurrentOption
691       \else
692         \expandafter\expandafter\expandafter\gdef
693         \expandafter\expandafter\expandafter\@unusedoptionlist
694         \expandafter\expandafter\expandafter{%
695           \expandafter\@unusedoptionlist
696           \expandafter,\CurrentOption
697         }%
698       \fi
699     }{%
700       \toks\tw@\expandafter{%
701         \the\toks@\expandafter\tw@\expandafter,\CurrentOption
702       }%
703     }%
704   }{}%
705   \toks@\expandafter{%
706     \the\expandafter\toks@\expandafter,\CurrentOption
707   }%
708 }%
709 }%

```

```
710     \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.

```
711     \@ifundefined{KVO@default@\currname.\@currentx}{%
712         \toks@\expandafter\expandafter\expandafter{%
713             \expandafter\KVO@temp\the\toks@
714         }%
715     }{%
716         \edef\CurrentOption{\the\toks@}%
717         \toks@\expandafter{\KVO@temp}%
718         \@for\CurrentOption:=\CurrentOption\do{%
719             \@ifundefined{%
720                 KV@#1@\expandafter\KVO@getkey\CurrentOption=\@nil
721             }{%
722                 \toks\tw@\expandafter{%
723                     \the\toks\expandafter\tw@\expandafter,\CurrentOption
724                 }%
725             }{%
726                 \toks@\expandafter{%
727                     \the\expandafter\toks@\expandafter,\CurrentOption
728                 }%
729             }%
730         }%
731     }%
732     \fi
733 }%
734 \edef\KVO@temp{\endgroup
735     \noexpand\KVO@calldefault{\the\toks\tw@}%
736     \noexpand\setkeys{#1}{\the\toks@}%
737 }%
738 \KVO@temp
```

Some cleanup of `\ProcessOptions`.

```
739 \let\CurrentOption\@empty
740 \AtEndOfPackage{\let\@unprocessedoptions\relax}%
741 }
```

### 6.5.2 `\ProcessLocalKeyvalOptions`

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

```
742 \newcommand*{\ProcessLocalKeyvalOptions}{%
743     \@ifstar{%
744         \begingroup
745         \edef\x{\endgroup
746             \noexpand\KVO@ProcessLocalKeyvalOptions{\KVO@family}%
747         }%
748         \x
749     }%
750     \KVO@ProcessLocalKeyvalOptions
751 }
```

```
752 \def\KVO@ProcessLocalKeyvalOptions#1{%
```

```
753     \let\@tempc\relax
754     \let\KVO@temp\@empty
```

Check if `\ProcessLocalKeyvalOptions` is called inside a package.

```
755     \ifx\@currentx\@pkgextension
756     \else
757         \PackageError{kvoptions}{%
758             \string\ProcessLocalKeyvalOptions\space is intended for packages only%
759         }\@ehc
760     \fi
```

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

```

761 \beginingroup
762   \toks\tw@{ }%
763   \@ifundefined{opt@\currname.\@currentt}{%
764     \toks@\expandafter{\KV0@temp}%
765   }{%
766     \toks@\expandafter\expandafter\expandafter{%
767       \csname opt@\currname.\@currentt\endcsname
768     }%

```

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.

```

769   \@ifundefined{KV0@default@\currname.\@currentt}{%
770     \toks@\expandafter\expandafter\expandafter{%
771       \expandafter\KV0@temp\the\toks@
772     }%
773   }{%
774     \edef\CurrentOption{\the\toks@}%
775     \toks@\expandafter{\KV0@temp}%
776     \@for\CurrentOption:=\CurrentOption\do{%
777       \@ifundefined{%
778         KV@#1@\expandafter\KV0@getkey\CurrentOption=\@nil
779       }{%
780         \toks\tw@\expandafter{%
781           \the\toks\expandafter\tw@\expandafter,\CurrentOption
782         }%
783       }{%
784         \toks@\expandafter{%
785           \the\expandafter\toks@\expandafter,\CurrentOption
786         }%
787       }%
788     }%
789   }%
790 }%
791 \edef\KV0@temp{\endgroup
792   \noexpand\KV0@calldefault{\the\toks\tw@}%
793   \noexpand\setkeys{#1}{\the\toks@}%
794 }%
795 \KV0@temp

```

Some cleanup of \ProcessOptions.

```

796 \let\CurrentOption\@empty
797 \AtEndOfPackage{\let\unprocessedoptions\relax}%
798 }

```

### 6.5.3 Helper macros

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

```

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

```

\KV0@calldefault

```

800 \def\KV0@calldefault#1{%
801   \beginingroup
802     \def\x{#1}%
803     \expandafter\endgroup
804     \ifx\x\@empty
805     \else
806       \@for\CurrentOption:=#1\do{%
807         \ifx\CurrentOption\@empty
808         \else
809           \expandafter\KV0@setcurrents\CurrentOption=\@nil

```

```

810     \nameuse{KVO@default@\currname.\@currentx}%
811     \fi
812   }%
813 \fi
814 }

```

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

```

815 \def\KVO@setcurrents#1=#2\@nil{%
816   \def\CurrentOptionValue{#2}%
817   \ifx\CurrentOptionValue\@empty
818     \let\CurrentOptionKey\CurrentOption
819     \let\CurrentOptionValue\relax
820   \else
821     \edef\CurrentOptionKey{\zap@space#1 \@empty}%
822     \expandafter\KVO@setcurrentvalue\CurrentOption\@nil
823   \fi
824 }

```

`\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.

```

825 \def\KVO@setcurrentvalue#1=#2\@nil{%
826   \KV@@sp@def\CurrentOptionValue{#2}%
827 }

```

## 6.6 plain- $\TeX$

Disable  $\LaTeX$  stuff.

```

828 \begingroup\expandafter\expandafter\expandafter\endgroup
829 \expandafter\ifx\csname documentclass\endcsname\relax
830   \def\ProcessKeyvalOptions{%
831     \@ifstar{}\@gobble
832   }%
833 \fi

834 \KVO@AtEnd
835 \</package>

```

## 6.7 Package `kvoptions-patch`

```

836 \<*patch>
837 \NeedsTeXFormat{LaTeX2e}
838 \begingroup
839   \catcode123 1 % {
840   \catcode125 2 % }
841   \def\x{\endgroup
842     \expandafter\edef\csname KVO@AtEnd\endcsname{%
843       \catcode35 \the\catcode35\relax
844       \catcode64 \the\catcode64\relax
845       \catcode123 \the\catcode123\relax
846       \catcode125 \the\catcode125\relax
847     }%
848   }%
849 \x
850 \catcode35 6 % #
851 \catcode64 11 % @
852 \catcode123 1 % {
853 \catcode125 2 % }
854 \def\TMP@EnsureCode#1#2{%
855   \edef\KVO@AtEnd{%
856     \KVO@AtEnd
857     \catcode#1 \the\catcode#1\relax
858   }%

```

```

859 \catcode#1 #2\relax
860 }
861 \TMP@EnsureCode{39}{12}% '
862 \TMP@EnsureCode{40}{12}% (
863 \TMP@EnsureCode{41}{12}% )
864 \TMP@EnsureCode{43}{12}% +
865 \TMP@EnsureCode{44}{12}% ,
866 \TMP@EnsureCode{45}{12}% -
867 \TMP@EnsureCode{46}{12}% .
868 \TMP@EnsureCode{47}{12}% /
869 \TMP@EnsureCode{58}{12}% :
870 \TMP@EnsureCode{60}{12}% <
871 \TMP@EnsureCode{61}{12}% =
872 \TMP@EnsureCode{62}{12}% >
873 \TMP@EnsureCode{91}{12}% [
874 \TMP@EnsureCode{93}{12}% ]
875 \TMP@EnsureCode{96}{12}% '
876 \TMP@EnsureCode{124}{12}% |
877 \edef\KVO@AtEnd{%
878 \KVO@AtEnd
879 \noexpand\endinput
880 }
881 \ProvidesPackage{kvoptions-patch}%
882 [2010/02/22 v3.7 LaTeX patch for keyval options (HO)]%

Check for  $\varepsilon$ -TeX.
883 \begingroup\expandafter\expandafter\expandafter\endgroup
884 \expandafter\ifx\csname eTeXversion\endcsname\relax
885 \PackageWarningNoLine{kvoptions-patch}{%
886 Package loading is aborted, because e-TeX is missing%
887 }%
888 \expandafter\KVO@AtEnd
889 \fi

Package etexcmds for \etex@unexpanded.
890 \RequirePackage{etexcmds}[2007/09/09]
891 \ifetex@unexpanded
892 \else
893 \PackageError{kvoptions-patch}{%
894 Could not find eTeX's \string\unexpanded.\MessageBreak
895 Try adding \string\RequirePackage\string{etexcmds\string} %
896 before \string\documentclass%
897 }\@ehd
898 \expandafter\KVO@AtEnd
899 \fi

Check for package xkvltxp.
900 \@ifpackageloaded{xkvltxp}{%
901 \PackageWarningNoLine{kvoptions}{%
902 Option 'patch' cannot be used together with\MessageBreak
903 package 'xkvltxp' that is already loaded.\MessageBreak
904 Therefore package loading is aborted%
905 }%
906 \KVO@AtEnd
907 }{}

908 \def\@if@ptions#1#2#3{%
909 \begingroup
910 \KVO@normalize\KVO@temp{#3}%
911 \edef\x{\endgroup
912 \noexpand\@if@ptions{%
913 \detokenize\expandafter\expandafter\expandafter{%
914 \csname opt@#2.#1\endcsname
915 }%
916 }{%

```

```

917     \detokenize\expandafter{\KVO@temp}%
918   }%
919 }%
920 \x
921 }

922 \def\@pass@ptions#1#2#3{%
923   \KVO@normalize\KVO@temp{#2}%
924   \@ifundefined{opt@#3.#1}{%
925     \expandafter\gdef\csname opt@#3.#1%
926       \expandafter\endcsname\expandafter{%
927         \KVO@temp
928       }%
929   }{%
930     \expandafter\gdef\csname opt@#3.#1%
931       \expandafter\expandafter\expandafter\endcsname
932       \expandafter\expandafter\expandafter{%
933         \csname opt@#3.#1\expandafter\endcsname\expandafter,\KVO@temp
934       }%
935   }%
936 }

937 \def\ProcessOptions{%
938   \let\ds@\@empty
939   \@ifundefined{opt@\@currname.\@currentx}{%
940     \let\@curroptions\@empty
941   }{%
942     \expandafter\expandafter\expandafter\def
943     \expandafter\expandafter\expandafter\@curroptions
944     \expandafter\expandafter\expandafter{%
945       \csname opt@\@currname.\@currentx\endcsname
946     }%
947   }%
948   \ifstar\KVO@xprocess@ptions\KVO@process@ptions
949 }

950 \def\KVO@process@ptions{%
951   \@for\CurrentOption:=\@declaredoptions\do{%
952     \ifx\CurrentOption\@empty
953     \else
954       \begingroup
955         \ifx\@currentx\@clsextension
956           \toks@{}%
957         \else
958           \toks@\expandafter{\@classoptionslist,}%
959         \fi
960         \toks\tw@\expandafter{\@curroptions}%
961         \edef\x{\endgroup
962           \noexpand\in@{\, \CurrentOption,}{, \the\toks@\the\toks\tw@,}%
963         }%
964       \x
965     \ifin@
966       \KVO@use@ption
967       \expandafter\let\csname ds@\CurrentOption\endcsname\@empty
968     \fi
969   \fi
970 }%
971 \KVO@process@ptions
972 }

973 \def\KVO@xprocess@ptions{%
974   \ifx\@currentx\@clsextension
975   \else
976     \@for\CurrentOption:=\@classoptionslist\do{%
977       \ifx\CurrentOption\@empty

```

```

978     \else
979     \KVO@in@\CurrentOption\@declaredoptions
980     \ifin@
981     \KVO@use@option
982     \expandafter\let\csname ds@\CurrentOption\endcsname\@empty
983     \fi
984     \fi
985     }%
986 \fi
987 \KVO@process@ptions
988 }

989 \def\KVO@in@#1#2{%
990 \in@false
991 \begingroup
992 \@for\x:=#2\do{%
993 \ifx\x#1\relax
994 \in@true
995 \fi
996 }%
997 \edef\x{\endgroup
998 \ifin@
999 \noexpand\in@true
1000 \fi
1001 }%
1002 \x
1003 }

1004 \def\KVO@process@ptions{%
1005 \@for\CurrentOption:=\@curroptions\do{%
1006 \@ifundefined{ds@\KVO@SanitizedCurrentOption}{%
1007 \KVO@use@option
1008 \default@ds
1009 }%
1010 \KVO@use@option
1011 }%
1012 \@for\CurrentOption:=\@declaredoptions\do{%
1013 \expandafter\let\csname ds@\CurrentOption\endcsname\relax
1014 }%
1015 \let\CurrentOption\@empty
1016 \let\@fileswith@ptions\@fileswith@ptions
1017 \AtEndOfPackage{\let\@unprocessedoptions\relax}%
1018 }

1019 \def\KVO@use@option{%
1020 \begingroup
1021 \edef\x{\endgroup
1022 \noexpand\@removeelement{%
1023 \detokenize\expandafter{\CurrentOption}%
1024 }{%
1025 \detokenize\expandafter{\@unusedoptionlist}%
1026 }%
1027 }%
1028 \x\@unusedoptionlist
1029 \csname ds@\KVO@SanitizedCurrentOption\endcsname
1030 }

1031 \def\OptionNotUsed{%
1032 \ifx\@current\@clsextension
1033 \xdef\@unusedoptionlist{%
1034 \ifx\@unusedoptionlist\@empty
1035 \else
1036 \detokenize\expandafter{\@unusedoptionlist,}%
1037 \fi
1038 \detokenize\expandafter{\CurrentOption}%

```

```

1039   }%
1040  \fi
1041 }

    Variant of \ExecuteOptions that better protects \CurrentOption.
1042 \def\CurrentOption@SaveLevel{0}
1043 \def\ExecuteOptions{%
1044   \expandafter\KVO@ExecuteOptions
1045     \csname CurrentOption@\CurrentOption@SaveLevel\endcsname
1046 }
1047 \def\KVO@ExecuteOptions#1#2{%
1048   \let#1\CurrentOption
1049   \edef\CurrentOption@SaveLevel{%
1050     \the\numexpr\CurrentOption@SaveLevel+1%
1051   }%
1052   \@for\CurrentOption:=#2\do{%
1053     \csname ds@\CurrentOption\endcsname
1054   }%
1055   \edef\CurrentOption@SaveLevel{%
1056     \the\numexpr\CurrentOption@SaveLevel-1%
1057   }%
1058   \let\CurrentOption#1%
1059 }

1060 \def\KVO@fileswith@ptions#1[#2]#3[#4]{%
1061   \ifx#1\@clsextension
1062     \ifx\@classoptionslist\relax
1063       \KVO@normalize\KVO@temp{#2}%
1064       \expandafter\gdef\expandafter\@classoptionslist\expandafter{%
1065         \KVO@temp
1066       }%
1067       \def\reserved@a{%
1068         \KVO@onefilewithoptions{#3}[{#2}][{#4}]#1%
1069         \@documentclasshook
1070       }%
1071     \else
1072       \def\reserved@a{%
1073         \KVO@onefilewithoptions{#3}[{#2}][{#4}]#1%
1074       }%
1075     \fi
1076   \else
1077     \begingroup
1078     \let\KVO@temp\relax
1079     \let\KVO@onefilewithoptions\relax
1080     \let\@pkgextension\relax
1081     \def\reserved@b##1,{%
1082       \ifx\@nil##1\relax
1083       \else
1084         \ifx\relax##1\relax
1085         \else
1086           \KVO@onefilewithoptions{##1}[{\KVO@temp}][{#4}]%
1087           \@pkgextension
1088         \fi
1089         \expandafter\reserved@b
1090       \fi
1091     }%
1092     \edef\reserved@a{\zap@space#3 \@empty}%
1093     \edef\reserved@a{\expandafter\reserved@b\reserved@a,\@nil,}%
1094     \toks@{#2}%
1095     \def\KVO@temp{\the\toks@}%
1096     \edef\reserved@a{\endgroup \reserved@a}%
1097   \fi
1098   \reserved@a
1099 }

```



```

1100 \def\KV0@onefilewithoptions#1[#2][#3]#4{%
1101   \@pushfilename
1102   \xdef\@currname{#1}%
1103   \global\let\@currentx#4%
1104   \expandafter\let\csname\@currname.\@currentx-h@k\endcsname\@empty
1105   \let\CurrentOption\@empty
1106   \@resetoptions
1107   \makeatletter
1108   \def\reserved@a{%
1109     \@ifl@aded\@currentx{#1}{%
1110       \@ifoptions\@currentx{#1}{#2}{%
1111         }{%
1112         \begingroup
1113         \@ifundefined{opt@#1.\@currentx}{%
1114           \def\x{%
1115             }{%
1116             \edef\x{%
1117               \expandafter\expandafter\expandafter\strip@prefix
1118               \expandafter\meaning\csname opt@#1.\@currentx\endcsname
1119             }%
1120             }%
1121             \def\y{#2}%
1122             \edef\y{\expandafter\strip@prefix\meaning\y}%
1123             \@latex@error{Option clash for \cls@pkg\space #1}{%
1124               The package #1 has already been loaded %
1125               with options:\MessageBreak
1126               \space\space[\x]\MessageBreak
1127               There has now been an attempt to load it %
1128               with options:\MessageBreak
1129               \space\space[\y]\MessageBreak
1130               Adding the global options:\MessageBreak
1131               \space\space
1132               \x,\y\MessageBreak
1133               to your \noexpand\documentclass declaration may fix this.%
1134               \MessageBreak
1135               Try typing \space <return> \space to proceed.%
1136             }%
1137             \endgroup
1138           }%
1139         }{%
1140           \@pass@options\@currentx{#2}{#1}%
1141           \global\expandafter
1142           \let\csname ver@\@currname.\@currentx\endcsname\@empty
1143           \InputIfFileExists
1144             {\@currname.\@currentx}%
1145             {}%
1146             {\@missingfileerror\@currname\@currentx}%
1147           \let\@unprocessedoptions\@unprocessedoptions
1148           \csname\@currname.\@currentx-h@k\endcsname
1149           \expandafter\let\csname\@currname.\@currentx-h@k\endcsname
1150             \undefined
1151           \@unprocessedoptions
1152         }%
1153         \@ifl@ter\@currentx{#1}{#3}{%
1154         }{%
1155         \@latex@warning@no@line{%
1156           You have requested,\on@line, %
1157           version\MessageBreak
1158           #3' of \cls@pkg\space #1,\MessageBreak
1159           but only version\MessageBreak
1160           '\csname ver@#1.\@currentx\endcsname'\MessageBreak
1161         is available%

```

```

1162     }%
1163 }%
1164 \ifx\@current\@clsextension\let\LoadClass\@twoloadclasserror\fi
1165 \@popfilename
1166 \@resetoptions
1167 }%
1168 \reserved@a
1169 }

1170 \def\@unknownoptionerror{%
1171 \@latexerror{%
1172   Unknown option '\KVO@SanitizedCurrentOption' %
1173   for \@cls@pkg\space'\@currname'%
1174 }{%
1175   The option '\KVO@SanitizedCurrentOption' was not declared in %
1176   \@cls@pkg\space'\@currname', perhaps you\MessageBreak
1177   misspelled its name. %
1178   Try typing \space <return> %
1179   \space to proceed.%
1180 }%
1181 }

1182 \def\@unprocessedoptions{%
1183 \ifx\@current\@pkgextension
1184 \ifundefined{opt@\@currname.\@current}{%
1185 \let\@curroptions\@empty
1186 }{%
1187 \expandafter\let\expandafter\@curroptions
1188 \csname opt@\@currname.\@current\endcsname
1189 }%
1190 \@for\CurrentOption:=\@curroptions\do{%
1191 \ifx\CurrentOption\@empty\else\@unknownoptionerror\fi
1192 }%
1193 \fi
1194 }

1195 \def\KVO@SanitizedCurrentOption{%
1196 \expandafter\strip@prefix\meaning\CurrentOption
1197 }

    Normalize option list.
1198 \def\KVO@normalize#1#2{%
1199 \let\KVO@result\@empty
1200 \KVO@splitcomma#2,\@nil
1201 \let#1\KVO@result
1202 }
1203 \def\KVO@splitcomma#1,#2\@nil{%
1204 \KVO@ifempty{#1}{}%
1205 \KVO@checkkv#1=\@nil
1206 }%
1207 \KVO@ifempty{#2}{\KVO@splitcomma#2\@nil}%
1208 }
1209 \def\KVO@ifempty#1{%
1210 \expandafter\ifx\expandafter\\detokenize{#1}\\%
1211 \expandafter\@firstoftwo
1212 \else
1213 \expandafter\@secondoftwo
1214 \fi
1215 }
1216 \def\KVO@checkkv#1=#2\@nil{%
1217 \KVO@ifempty{#2}{%
1218 % option without value
1219 \edef\KVO@x{\zap@space#1 \@empty}%
1220 \ifx\KVO@x\@empty
1221 % ignore empty option

```

```

1222 \else
1223 % append to list
1224 \edef\KVO@result{%
1225 \etex@unexpanded\expandafter{\KVO@result},\KVO@x
1226 }%
1227 \fi
1228 }{%
1229 % #1: "key", #2: "value="
1230 % add key part
1231 \edef\KVO@result{%
1232 \etex@unexpanded\expandafter{\KVO@result},%
1233 \zap@space#1 \@empty
1234 }%
1235 \futurelet\@let@token\KVO@checkfirsttok#2 \@nil = \@nil|\KVO@nil
1236 }%
1237 }
1238 \def\KVO@checkfirsttok{%
1239 \ifx\@let@token\bgroup
1240 % no space at start
1241 \expandafter\KVO@removelastspace\expandafter=%
1242 % "<value><spaceopt>= \@nil"
1243 \else
1244 \expandafter\KVO@checkfirstA
1245 \fi
1246 }
1247 \def\KVO@checkfirstA#1 #2\@nil{%
1248 \KVO@ifempty{#2}{%
1249 \KVO@removelastspace=#1 \@nil
1250 }{%
1251 \KVO@ifempty{#1}{%
1252 \KVO@removelastspace=#2\@nil
1253 }{%
1254 \KVO@removelastspace=#1 #2\@nil
1255 }%
1256 }%
1257 }
1258 \def\KVO@removelastspace#1 = \@nil|#2\KVO@nil{%
1259 \KVO@ifempty{#2}{%
1260 \edef\KVO@result{%
1261 \etex@unexpanded\expandafter{\KVO@result}%
1262 \etex@unexpanded\expandafter{\KVO@removegarbage#1\KVO@nil}%
1263 }%
1264 }{%
1265 \edef\KVO@result{%
1266 \etex@unexpanded\expandafter{\KVO@result}%
1267 \etex@unexpanded{#1}%
1268 }%
1269 }%
1270 }
1271 \def\KVO@removegarbage#1= \@nil#2\KVO@nil{#1}%

```

Arguments #1 and #2 are macros.

```

1272 \def\KVO@removeelement#1#2{%
1273 \begingroup
1274 \toks@={}%
1275 \for\x:=#2\do{%
1276 \ifx\x\@empty
1277 \else
1278 \ifx\x#1\relax
1279 \else
1280 \edef\t{\the\toks@}%
1281 \ifx\t\@empty
1282 \else

```

```

1283         \toks@\expandafter{\the\toks@,}%
1284     \fi
1285     \toks@\expandafter{\the\expandafter\toks@\x}%
1286 \fi
1287 \fi
1288 }%
1289 \edef\x{\endgroup
1290     \def\noexpand#2{\the\toks@}%
1291 }%
1292 \x
1293 }

1294 \let\@@fileswith@pti@ns\KVO@fileswith@pti@ns
1295 \ifx\@fileswith@pti@ns\@badrequireerror
1296 \else
1297     \let\@fileswith@pti@ns\KVO@fileswith@pti@ns
1298 \fi

```

\KVO@Patch

```

1299 \let\KVO@Patch=Y

1300 \KVO@AtEnd
1301 \</patch>

```

## 7 Test

### 7.1 Preface for standard catcode check

```

1302 <*test1>
1303 \input miniltx.tex\relax
1304 </test1>

```

### 7.2 Catcode checks for loading

```

1305 <*test1>

1306 \catcode'\{=1 %
1307 \catcode'\}=2 %
1308 \catcode'\#=6 %
1309 \catcode'\@=11 %
1310 \expandafter\ifx\csname count@\endcsname\relax
1311     \countdef\count@=255 %
1312 \fi
1313 \expandafter\ifx\csname @gobble\endcsname\relax
1314     \long\def\@gobble#1{}%
1315 \fi
1316 \expandafter\ifx\csname @firstofone\endcsname\relax
1317     \long\def\@firstofone#1{#1}%
1318 \fi
1319 \expandafter\ifx\csname loop\endcsname\relax
1320     \expandafter\@firstofone
1321 \else
1322     \expandafter\@gobble
1323 \fi
1324 {%
1325     \def\loop#1\repeat{%
1326         \def\body{#1}%
1327         \iterate
1328     }%
1329     \def\iterate{%
1330         \body
1331         \let\next\iterate
1332     \else
1333         \let\next\relax

```

```

1334 \fi
1335 \next
1336 }%
1337 \let\repeat=\fi
1338 }%
1339 \def\RestoreCatcodes{}
1340 \count@=0 %
1341 \loop
1342 \edef\RestoreCatcodes{%
1343 \RestoreCatcodes
1344 \catcode\the\count@=\the\catcode\count@\relax
1345 }%
1346 \ifnum\count@<255 %
1347 \advance\count@ 1 %
1348 \repeat
1349
1350 \def\RangeCatcodeInvalid#1#2{%
1351 \count@=#1\relax
1352 \loop
1353 \catcode\count@=15 %
1354 \ifnum\count@<#2\relax
1355 \advance\count@ 1 %
1356 \repeat
1357 }
1358 \expandafter\ifx\csname LoadCommand\endcsname\relax
1359 \def\LoadCommand{\input kvoptions.sty\relax}%
1360 \fi
1361 \def\Test{%
1362 \RangeCatcodeInvalid{0}{47}%
1363 \RangeCatcodeInvalid{58}{64}%
1364 \RangeCatcodeInvalid{91}{96}%
1365 \RangeCatcodeInvalid{123}{255}%
1366 \catcode'\@=12 %
1367 \catcode'\=0 %
1368 \catcode'\{=1 %
1369 \catcode'\}=2 %
1370 \catcode'\#=6 %
1371 \catcode'\[=12 %
1372 \catcode'\]=12 %
1373 \catcode'\%=14 %
1374 \catcode'\ =10 %
1375 \catcode13=5 %
1376 \LoadCommand
1377 \RestoreCatcodes
1378 }
1379 \Test
1380 \csname @@end\endcsname
1381 \end

1382 </test1>

1383 <*test2>
1384 \NeedsTeXFormat{LaTeX2e}
1385 \makeatletter
1386 \catcode'\@=11 %
1387 \def\RestoreCatcodes{}
1388 \count@=0 %
1389 \loop
1390 \edef\RestoreCatcodes{%
1391 \RestoreCatcodes
1392 \catcode\the\count@=\the\catcode\count@\relax
1393 }%
1394 \ifnum\count@<255 %
1395 \advance\count@\@ne

```

```

1396 \repeat
1397
1398 \def\RangeCatcodeInvalid#1#2{%
1399   \count@=#1\relax
1400   \loop
1401     \catcode\count@=15 %
1402   \ifnum\count@<#2\relax
1403     \advance\count@\@ne
1404   \repeat
1405 }
1406 \def\Test#1{%
1407   \RangeCatcodeInvalid{0}{47}%
1408   \RangeCatcodeInvalid{58}{64}%
1409   \RangeCatcodeInvalid{91}{96}%
1410   \RangeCatcodeInvalid{123}{255}%
1411   \catcode'\@=12 %
1412   \catcode'\=0 %
1413   \catcode'\{=1 %
1414   \catcode'\}=2 %
1415   \catcode'\#=6 %
1416   \catcode'\[=12 %
1417   \catcode'\]=12 %
1418   \catcode'\%=14 %
1419   \catcode'\ =10 %
1420   \catcode13=5 %
1421   #1\relax
1422   \RestoreCatcodes
1423 }
1424 \Test{\RequirePackage{kvoptions-patch}}%
1425 \Test{\RequirePackage{kvoptions}}%
1426 \csname @@end\endcsname
1427 </test2>
1428 <*test3>
1429 \NeedsTeXFormat{LaTeX2e}
1430 \makeatletter
1431 \RequirePackage{kvoptions}[2010/02/22]
1432 \def\msg#\{\immediate\write16}
1433 \define@key{testfamily}{testkey}{%
1434   \msg{[testfamily/testkey/#1]}%
1435 }
1436 \define@key{testfamily}{testdefaultkey}[testdefault]{%
1437   \msg{[testfamily/testdefaultkey/#1]}%
1438 }
1439 \AddToKeyvalOption{testfamily}{testkey}{%
1440   \msg{[addition/#1]}%
1441 }
1442 \AddToKeyvalOption{testfamily}{testdefaultkey}{%
1443   \msg{[addition/#1]}%
1444 }
1445 \setkeys{testfamily}{%
1446   testkey=testA,%
1447   testdefaultkey=testB,%
1448   testdefaultkey,%
1449 }
1450 \SetupKeyvalOptions{%
1451   family=testfamily%
1452 }
1453 \AddToKeyvalOption*{testkey}{%
1454   \msg{[star addition/#1]}%
1455 }
1456 \AddToKeyvalOption*{testdefaultkey}{%
1457   \msg{[star addition/#1]}%

```

```

1458 }
1459 \setkeys{testfamily}{%
1460   testkey=testA,%
1461   testdefaultkey=testB,%
1462   testdefaultkey,%
1463 }
1464 \@@end
1465 </test3>

```

## 8 Installation

### 8.1 Download

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

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

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

**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):

```

kvoptions.sty           → tex/latex/oberdiek/kvoptions.sty
kvoptions-patch.sty    → tex/latex/oberdiek/kvoptions-patch.sty
kvoptions.pdf          → doc/latex/oberdiek/kvoptions.pdf
example-mycolorsetup.sty → doc/latex/oberdiek/example-mycolorsetup.sty
test/kvoptions-test1.tex → doc/latex/oberdiek/test/kvoptions-test1.tex
test/kvoptions-test2.tex → doc/latex/oberdiek/test/kvoptions-test2.tex
test/kvoptions-test3.tex → doc/latex/oberdiek/test/kvoptions-test3.tex
kvoptions.dtx          → source/latex/oberdiek/kvoptions.dtx

```

---

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

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  $\TeX$  distribution (`teTeX`, `mikTeX`, ...) relies on file name databases, you must refresh these. For example, `teTeX` users run `texhash` or `mktexlsr`.

## 8.5 Some details for the interested

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

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

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

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

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

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

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

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

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

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

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

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

## 9 References

- [1] 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 `tracefmt`, Frank Mittelbach, Rainer Schöpf, 1997/05/29. [CTAN:macros/latex/base/lfsstrc.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](#)

## 10 History

### [0000/00/00 v0.0]

- Probably David Carlisle’s code in `hyperref` was the start.

### [2004/02/22 v1.0]

- The first version was never published. It also has offered a patch to get rid of L<sup>A</sup>T<sub>E</sub>X’s option expansion.

### [2006/02/16 v2.0]

- Now the package is redesigned with an easier user interface.
- `\ProcessKeyvalOptions` remains the central service, inherited from `hyperref`’s `\ProcessOptionsWithKV`. Now the use inside classes is also supported.
- Provides help macros for boolean and simple string options.
- Fixes for the patch of L<sup>A</sup>T<sub>E</sub>X. The patch is only enabled, if the user requests it.

### [2006/02/20 v2.1]

- Unused option list is sanitized to prevent problems with other packages that uses own processing methods for key value options. Disadvantage: the unused global option detection is weakened.
- New option type by `\DeclareVoidOption` for options without value.
- Default rule by `\DeclareDefaultOption`.
- Dynamic options: `\DisableKeyvalOption`.

### [2006/06/01 v2.2]

- Fixes for option patch.

### [2006/08/17 v2.3]

- `\DeclareBooleanOption` renamed to `\DeclareBoolOption` to avoid a name clash with package `\ifoption`.

[2006/08/22 v2.4]

- Option patch: `\ExecuteOptions` does not change the meaning of macro `\CurrentOption` at all.

[2007/04/11 v2.5]

- Line ends sanitized.

[2007/05/06 v2.6]

- Uses package `etexcmds`.

[2007/06/11 v2.7]

- The patch part fixes LaTeX bug `latex/3965`.

[2007/10/02 v2.8]

- Compatibility for plain-TeX 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.

- Missing space in error message added.

## 11 Index

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

Symbols	
<code>\#</code> .....	1308, 1370, 1415
<code>\%</code> .....	1373, 1418
<code>\@</code> .....	1309, 1366, 1386, 1411
<code>\@@end</code> .....	1464
<code>\@@fileswith@pti@ns</code> .....	1016, 1294
<code>\@@unprocessedoptions</code> .....	1147, 1182
<code>\@SetupDriver</code> .....	41, 43
<code>\@badrequireerror</code> .....	1295
<code>\@classoptionslist</code> .....	...
	635, 637, 958, 976, 1062, 1064
<code>\@cls@pkg</code> .....	1123, 1158, 1173, 1176
<code>\@clsextension</code> .....	274, 275, 294, 314,
	315, 339, 352, 385, 413, 414,
	435, 445, 472, 552, 564, 575,
	633, 678, 955, 974, 1032, 1061, 1164
<code>\@currrent</code> .....	221, 232,
	235, 238, 243, 246, 249, 274,
	294, 295, 314, 385, 413, 453,
	633, 672, 676, 678, 685, 711,
	755, 763, 767, 769, 810, 939,
	945, 955, 974, 1032, 1103, 1104,
	1109, 1110, 1113, 1118, 1140,
	1142, 1144, 1146, 1148, 1149,
	1153, 1160, 1164, 1183, 1184, 1188
<code>\@currname</code> .....	57, 218,
	232, 235, 236, 238, 243, 246,
	247, 249, 273, 295, 313, 390,
	412, 453, 672, 676, 685, 711,
	763, 767, 769, 810, 939, 945,
	1102, 1104, 1142, 1144, 1146,
	1148, 1149, 1173, 1176, 1184, 1188
<code>\@curroptions</code> .....	940,
	943, 960, 1005, 1185, 1187, 1190
<code>\@declaredoptions</code> .....	951, 979, 1012
<code>\@documentclasshook</code> .....	1069
<code>\@ehc</code> .....	301, 487, 536, 572, 759
<code>\@ehd</code> .....	897
<code>\@empty</code> .....	88, 89, 469, 518,
	632, 689, 739, 754, 796, 804,
	807, 817, 821, 938, 940, 952,
	967, 977, 982, 1015, 1034, 1092,
	1104, 1105, 1142, 1185, 1191,
	1199, 1219, 1220, 1233, 1276, 1281
<code>\@expandtwoargs</code> .....	663
<code>\@fileswith@pti@ns</code> ..	1016, 1295, 1297
<code>\@firstofone</code> ..	334, 337, 405, 1317, 1320
<code>\@firstoftwo</code> .....	1211
<code>\@for</code> 637, 681, 718, 776, 806, 951, 976,	
	992, 1005, 1012, 1052, 1190, 1275
<code>\@gobble</code> 348, 514, 601, 831, 1314, 1322	
<code>\@gobbletwo</code> .....	403
<code>\@if@pti@ns</code> .....	912
<code>\@if@ptions</code> .....	908, 1110
<code>\@ifdefinable</code> .....	328
<code>\@ifl@aded</code> .....	1109
<code>\@ifl@ter</code> .....	1153
<code>\@ifnextchar</code> .....	366
<code>\@ifpackageloaded</code> .....	900
<code>\@ifstar</code> .....	586, 621, 743, 831, 948
<code>\@ifundefined</code> .....	199, 203,
	208, 211, 217, 220, 235, 246,
	261, 288, 482, 596, 638, 641,
	672, 682, 685, 711, 719, 763,
	769, 777, 924, 939, 1006, 1113, 1184
<code>\@latex@error</code> .....	1123, 1171
<code>\@latex@warning@no@line</code> .....	1155
<code>\@let@token</code> .....	1235, 1239
<code>\@missingfileerror</code> .....	1146
<code>\@namedef</code> .....	374, 424, 453
<code>\@nameuse</code> .....	810
<code>\@ne</code> .....	223, 1395, 1403
<code>\@nil</code> .....	639, 642, 683,
	720, 778, 799, 809, 815, 822,
	825, 1082, 1093, 1200, 1203,
	1205, 1207, 1216, 1235, 1242,
	1247, 1249, 1252, 1254, 1258, 1271
<code>\@onelevel@sanitize</code> .....	...
	204, 332, 363, 364, 654, 687
<code>\@pass@ptions</code> .....	922, 1140
<code>\@pkgextension</code> .....	277,
	317, 416, 476, 755, 1080, 1087, 1183
<code>\@popfilename</code> .....	1165
<code>\@pushfilename</code> .....	1101
<code>\@removeelement</code> .....	663, 1022
<code>\@reset@ptions</code> .....	1106, 1166
<code>\@secondoftwo</code> .....	1213
<code>\@tempc</code> .....	631, 753
<code>\@twoloadclasserror</code> .....	1164
<code>\@typeset@protect</code> .....	215
<code>\@undefined</code> .....	149, 1150
<code>\@unknownoptionerror</code> ..	68, 1170, 1191
<code>\@unprocessedoptions</code> .....	...
	740, 797, 1017, 1147, 1151
<code>\@unusedoptionlist</code> .....	...
	664, 689, 690, 693,
	695, 1025, 1028, 1033, 1034, 1036
<code>\@x@protect</code> .....	212
<code>\[</code> .....	1371, 1416
<code>\]</code> .....	376, 1210, 1367, 1412
<code>\{</code> .....	1306, 1368, 1413
<code>\}</code> .....	1307, 1369, 1414

<code>\]</code> .....	1372, 1417	<code>\CurrentOptionKey</code> .....	818, 821
<code>\_</code> .....	1374, 1419	<code>\CurrentOptionValue</code> .....	56, 816, 817, 819, 826
<b>A</b>			
<code>\AddToKeyvalOption</code> .....	8, 585, 599, 1439, 1442, 1453, 1456	<code>\DeclareBoolOption</code> .	5, 20, 22, 255, 465
<code>\advance</code> .....	1347, 1355, 1395, 1403	<code>\DeclareComplementaryOption</code> ....	6, 287, 466
<code>\afterassignment</code> .....	607	<code>\DeclareDefaultOption</code> ....	6, 55, 452
<code>\aftergroup</code> .....	123	<code>\DeclareLocalOption</code> .....	7
<code>\AtEndOfPackage</code> ..	225, 740, 797, 1017	<code>\DeclareLocalOptions</code> .....	455
<b>B</b>			
<code>\body</code> .....	1326, 1330	<code>\DeclareOption</code> .....	223, 224
<b>C</b>			
<code>\catcode</code> .....	100, 101, 102, 103, 104, 105, 106, 114, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 161, 162, 165, 166, 167, 168, 172, 173, 174, 175, 179, 181, 223, 839, 840, 843, 844, 845, 846, 850, 851, 852, 853, 857, 859, 1306, 1307, 1308, 1309, 1344, 1353, 1366, 1367, 1368, 1369, 1370, 1371, 1372, 1373, 1374, 1375, 1386, 1392, 1401, 1411, 1412, 1413, 1414, 1415, 1416, 1417, 1418, 1419, 1420	<code>\DeclareStringOption</code> .....	4, 26, 28, 50, 52, 365, 467
<code>\ClassError</code> .....	565	<code>\DeclareVoidOption</code> ..	6, 32, 33, 34, 401
<code>\ClassInfo</code> .....	353, 386, 446, 553	<code>\default@ds</code> .....	1008
<code>\ClassWarning</code> .....	340, 436, 576	<code>\define@key</code> ....	230, 241, 272, 312, 382, 411, 470, 474, 539, 1433, 1436
<code>\comma@parse</code> .....	456	<code>\detokenize</code> .....	913, 917, 1023, 1025, 1036, 1038, 1210
<code>\count@</code> .....	1311, 1340, 1344, 1346, 1347, 1351, 1353, 1354, 1355, 1388, 1392, 1394, 1395, 1399, 1401, 1402, 1403	<code>\DisableKeyvalOption</code> .....	7, 478
<code>\countdef</code> .....	1311	<code>\do</code>	637, 681, 718, 776, 806, 951, 976, 992, 1005, 1012, 1052, 1190, 1275
<code>\csname</code> .....	107, 115, 141, 157, 164, 231, 238, 242, 249, 259, 260, 268, 306, 307, 308, 309, 328, 358, 395, 420, 459, 489, 497, 501, 513, 517, 540, 545, 546, 548, 549, 605, 676, 767, 829, 842, 884, 914, 925, 930, 933, 945, 967, 982, 1013, 1029, 1045, 1053, 1104, 1118, 1142, 1148, 1149, 1160, 1188, 1310, 1313, 1316, 1319, 1358, 1380, 1426	<code>\documentclass</code> .....	896, 1133
<code>\CurrentOption</code> .....	35, 37, 38, 41, 58, 62, 64, 429, 679, 681, 683, 687, 690, 696, 701, 706, 716, 718, 720, 723, 727, 739, 774, 776, 778, 781, 785, 796, 806, 807, 809, 818, 822, 951, 952, 962, 967, 976, 977, 979, 982, 1005, 1012, 1013, 1015, 1023, 1038, 1048, 1052, 1053, 1058, 1105, 1190, 1191, 1196	<code>\ds@</code> .....	938
<code>\CurrentOption@SaveLevel</code> .....	1042, 1045, 1049, 1050, 1055, 1056	<b>E</b>	
		<code>\emph</code> .....	48, 87, 91, 93
		<code>\empty</code> .....	110, 111
		<code>\end</code> .....	1381
		<code>\endcsname</code> ....	107, 115, 141, 157, 164, 232, 238, 243, 249, 259, 260, 268, 306, 307, 308, 309, 328, 358, 395, 420, 459, 489, 497, 501, 513, 517, 540, 545, 546, 548, 549, 605, 676, 767, 829, 842, 884, 914, 926, 931, 933, 945, 967, 982, 1013, 1029, 1045, 1053, 1104, 1118, 1142, 1148, 1149, 1160, 1188, 1310, 1313, 1316, 1319, 1358, 1380, 1426
		<code>\endinput</code> .....	123, 879
		<code>\etex@unexpanded</code> .....	645, 649, 1225, 1232, 1261, 1262, 1266, 1267
		<code>\ExecuteOptions</code> .....	1043
		<b>F</b>	
		<code>\futurelet</code> .....	1235
		<b>G</b>	
		<code>\gdef</code> .....	692, 925, 930, 1064
		<b>I</b>	
		<code>\ifetex@unexpanded</code> .....	891
		<code>\ifin@</code> .....	965, 980, 998
		<code>\ifKV@dyn@global</code> .....	495, 499, 506, 511, 515, 522, 543, 559
		<code>\ifMCS@print</code> .....	79
		<code>\ifnum</code> .....	1346, 1354, 1394, 1402
		<code>\ifpdf</code> .....	25
		<code>\ifx</code> ....	56, 88, 108, 111, 115, 141, 149, 152, 274, 294, 314, 333, 336, 339, 352, 376, 385, 413,



<b>P</b>		
\PackageError .....	679, 701, 706, 713, 716, 723,	
..... 289, 483, 533, 567, 757, 893	727, 735, 736, 771, 774, 781,	
\PackageInfo .....	785, 792, 793, 843, 844, 845,	
..... 120, 355, 388, 448, 504, 520, 555	846, 857, 962, 1050, 1056, 1095,	
\PackageWarning 262, 342, 438, 578, 597	1280, 1283, 1285, 1290, 1344, 1392	
\PackageWarningNoLine .. 57, 885, 901	\TMP@EnsureCode ... 176, 183, 184,	
\PassOptionsToPackage .....	185, 186, 187, 188, 189, 190,	
..... 63, 80	191, 192, 193, 194, 195, 196,	
\ProcessKeyvalOptions 3, 74, 620, 830	197, 198, 854, 861, 862, 863,	
\ProcessLocalKeyvalOptions 4, 742, 758	864, 865, 866, 867, 868, 869,	
\ProcessOptions .....	870, 871, 872, 873, 874, 875, 876	
..... 229, 937	\toks .....	
\protect .....	671, 700, 701, 722, 723,	
..... 213	735, 762, 780, 781, 792, 960, 962	
\ProvidesPackage .....	\toks@ .....	
..... 5, 112, 158, 881	377, 379, 382, 384,	
<b>R</b>		
\RangeCatcodeInvalid .....	391, 613, 614, 616, 673, 675,	
..... 1350, 1362, 1363, 1364,	679, 680, 705, 706, 712, 713,	
1365, 1398, 1407, 1408, 1409, 1410	716, 717, 726, 727, 736, 764,	
\renewcommand .....	766, 770, 771, 774, 775, 784,	
..... 93	785, 793, 956, 958, 962, 1094,	
\repeat 1325, 1337, 1348, 1356, 1396, 1404	1095, 1274, 1280, 1283, 1285, 1290	
\RequirePackage .....	\tw@ .....	
..... 7, 8, 84, 200,	671, 700, 701, 722, 723,	
202, 226, 890, 895, 1424, 1425, 1431	735, 762, 780, 781, 792, 960, 962	
\reserved@a .....	<b>U</b>	
..... 1067, 1072,	\unexpanded .....	
1092, 1093, 1096, 1098, 1108, 1168	..... 894	
\reserved@b .....	<b>W</b>	
..... 1081, 1089, 1093	\write .....	
\RestoreCatcodes .....	..... 117, 143, 1432	
..... 1343, 1377, 1387, 1390, 1391, 1422	<b>X</b>	
<b>S</b>		
\setkeys .....	\x .....	
..... 44, 253, 480, 736, 793, 1445, 1459	107, 108, 111, 116, 120,	
\SetupDriver .....	122, 142, 147, 157, 163, 171,	
..... 32, 33, 34, 35, 38, 40	271, 282, 311, 322, 381, 398,	
\SetupKeyvalOptions .....	410, 423, 431, 433, 481, 491,	
..... 4, 13, 252, 461, 1450	494, 510, 538, 588, 591, 615,	
\space .....	618, 623, 626, 745, 748, 802,	
..... 294,	804, 841, 849, 911, 920, 961,	
758, 1123, 1126, 1129, 1131,	964, 992, 993, 997, 1002, 1021,	
1135, 1158, 1173, 1176, 1178, 1179	1028, 1114, 1116, 1126, 1132,	
\strip@prefix .....	1275, 1276, 1278, 1285, 1289, 1292	
..... 205, 209, 485, 1117, 1122, 1196	<b>Y</b>	
<b>T</b>		
\t .....	\y .....	
..... 1280, 1281	1121, 1122, 1129, 1132	
\Test .....	<b>Z</b>	
..... 1361, 1379, 1406, 1424, 1425	\zap@space .....	
\textcolor .....	..... 821, 1092, 1219, 1233	
..... 94		
\the .....		
..... 165, 166, 167,		
168, 179, 382, 391, 614, 616,		