

The kvoptions package

Heiko Oberdiek*

2022-06-15 v3.15

Abstract

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

Contents

1	Introduction	3
1.1	The beginning	3
1.2	Overview	3
2	Usage	3
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>	5
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 \TeX	9
3	Example	9
4	Package options	11
4.1	Package kvoptions-patch	11
4.2	Option debugshow	12
5	Limitations	13
5.1	Compatibility	13
5.1.1	Package kvoptions-patch vs. package xkvltxp	13
5.2	Limitations	13
5.2.1	Option comparisons	13
5.2.2	Option list parsing with package kvoptions-patch	13

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

6	Implementation	14
6.1	Disabling the patches for newer LaTeX	14
6.2	Preamble	14
6.3	Option declaration macros	17
6.3.1	<code>\SetupKeyvalOptions</code>	17
6.3.2	<code>\DeclareBoolOption</code>	18
6.3.3	<code>\DeclareStringOption</code>	20
6.3.4	<code>\DeclareVoidOption</code>	21
6.3.5	<code>\DeclareDefaultOption</code>	22
6.3.6	<code>\DeclareLocalOptions</code>	22
6.4	Dynamic options	22
6.4.1	<code>\DisableKeyvalOption</code>	22
6.5	Change option code	24
6.5.1	<code>\AddToKeyvalOption</code>	24
6.6	Process options	25
6.6.1	Get global options	25
6.6.2	<code>\ProcessKeyvalOptions</code>	25
6.6.3	<code>\ProcessLocalKeyvalOptions</code>	28
6.6.4	Helper macros	29
6.7	plain \TeX	30
6.8	Package <code>kvoptions-patch</code>	30
7	Installation	38
7.1	Download	38
7.2	Package installation	38
7.3	Refresh file name databases	38
7.4	Some details for the interested	38
8	References	39
9	History	39
	[0000/00/00 v0.0]	39
	[2004/02/22 v1.0]	39
	[2006/02/16 v2.0]	39
	[2006/02/20 v2.1]	40
	[2006/06/01 v2.2]	40
	[2006/08/17 v2.3]	40
	[2006/08/22 v2.4]	40
	[2007/04/11 v2.5]	40
	[2007/05/06 v2.6]	40
	[2007/06/11 v2.7]	40
	[2007/10/02 v2.8]	40
	[2007/10/11 v2.9]	40
	[2007/10/18 v3.0]	40
	[2009/04/10 v3.1]	40
	[2009/07/17 v3.2]	40
	[2009/07/21 v3.3]	41
	[2009/08/13 v3.4]	41
	[2009/12/04 v3.5]	41
	[2009/12/08 v3.6]	41
	[2010/02/22 v3.7]	41
	[2010/07/23 v3.8]	41
	[2010/12/02 v3.9]	41
	[2010/12/23 v3.10]	41
	[2011/06/30 v3.11]	41
	[2016/05/16 v3.12]	41
	[2019/11/29 v3.13]	41
	[2020-10-07 v3.14]	41

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^AT_EX’s package and class `options`:

Package <code>keyval</code>	Package <code>kvoptions</code>	L ^A T _E X kernel
<code>\define@key</code>	<code>\DeclareVoidOption</code> <code>\DeclareStringOption</code> <code>\DeclareBoolOption</code> <code>\DeclareComplementaryOption</code> <code>\DisableKeyvalOption</code>	<code>\DeclareOption</code>
	<code>\DeclareDefaultOption</code>	<code>\DeclareOption*</code>
	<code>\ProcessKeyvalOptions</code>	<code>\ProcessOptions*</code>
	Option patch	Class/package option system
	<code>\SetupKeyvalOptions</code>	

2 Usage

2.1 Process options

2.1.1 `\ProcessKeyvalOptions`

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

This command evaluates the global or local options of the package that are defined with `keyval`’s interface within the family `<family>`. It acts the same way as L^AT_EX’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⟩
  setkeys = ⟨setkeys command⟩
}
```

This command allows to configure the default assumptions that are based on the current package or class name. L^AT_EX 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
<code>family</code>	<code>⟨\@currname⟩</code>	<code>(foobar)</code>	<code>\ProcessKeyvalOptions*</code> <code>\DeclareBoolOption</code> <code>\DeclareStringOption</code>
<code>prefix</code>	<code>⟨\@currname⟩@</code>	<code>(foobar@)</code>	<code>\DeclareBoolOption</code> <code>\DeclareStringOption</code> <code>\DeclareVoidOption</code>
<code>setkeys</code>	<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 `koptions` 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 `<key>` is defined. The option code just stores its value in the macro. If the optional argument at the end of `\DeclareStringOption` is given, then option `<key>` is defined with the default `<default>`.

Example for a package with the following two lines:

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

Then `\DeclareStringOption` defines the macro with content `me`, note \LaTeX complains if the name of the macro already exists:

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

The option definition is similar to:

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

2.2.2 `\DeclareBoolOption`

```
\DeclareBoolOption [init] {key}
```

A boolean switch is generated, initialized by value `<init>` and the corresponding key `<key>` is defined. If the initialization value is not given, `false` is used as default.

The internal actions of `\DeclareBoolOption` are shown below. The example is given for a package author who has the following two lines in his package/class:

```
\ProvidesPackage{foobar}
\DeclareBoolOption{verbose}
```

First a new switch is created:

```
\newif\iffoobar@verbose
```

and initialized:

```
\foobar@verbosefalse
```

Finally the key is defined:

```
\define@key{foobar}{verbose}[true]{...}
```

The option code configures the boolean option in the following way: If the author specifies `true` or `false` then the switch is turned on or off respectively. Also the option can be given without explicit value. Then the switch is enabled. Other values are reported as errors.

Now the switch is ready to use in the package/class, e.g.:

```

\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^AT_EX'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^AT_EX 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 \LaTeX 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 \LaTeX '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
  package or class = ⟨name⟩                             default: global
```

`\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`

<pre> \AddToKeyvalOption {<family>} {<key>} {<code>} \AddToKeyvalOption * {<key>} {<code>} </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 `\<prefix>` that can be configured by `\SetupKeyvalOptions`.

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

2.5 plain T_EX

Package `keyval` is also usable in plain T_EX with the help of file `miniltx.tex`. Some features of this package `kvoptions` might also be useful for plain T_EX. If L^AT_EX 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}[2019/11/29 Managing my colors]
6
7 \RequirePackage{iftex}
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^AT_EX 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`

Change in version v3.14: `kvoptions-patch` is not compatible with a L^AT_EX 2020-10-01 or newer and will abort loading if it detects it!

L^AT_EX'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^AT_EX 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^AT_EX 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 package `kvoptions-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{kvoptions-patch}
\documentclass[pdfauthor=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 ϵ - \TeX , 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 ϵ - \TeX .
- 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 L^AT_EX's kernel commands in order to get better support for key value options. Of course they cannot be used both. The user must decide, which method he prefers. Package `kvoptions-patch` aborts itself, if it detects that `xkvltxp` is already loaded.

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

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

The other way should work, too.

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

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

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

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

5.2 Limitations

5.2.1 Option comparisons

In some situations L^AT_EX compares option lists, e.g. option clash check, `\@ifpackagewith`, or `\@ifclasswith`. Apart from catcode and sanitizing problems of option `patch`, there is another problem. L^AT_EX 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 Disabling the patches for newer LaTeX

kvoptions-patch is not compatible with L^AT_EX 2020-10-01 and newer so it is disabled and issues a warning.

```
98 (*patch)
99 \providecommand\IfFormatAtLeastTF{\@ifl@t@r\fmtversion}
100 \IfFormatAtLeastTF{2020/10/01}{\PackageWarning{kvoptions-patch}%
101 {kvoptions-patch is not compatible with \MessageBreak
102 LaTeX \fmtversion\MessageBreak Loading is aborted}{}}{}
103 \IfFormatAtLeastTF{2020/10/01}{\endinput}{}
104 </patch>
```

6.2 Preamble

```
105 (*package)
```

Reload check and identification. Reload check, especially if the package is not used with L^AT_EX.

```
106 \begingroup\catcode61\catcode48\catcode32=10\relax%
107 \catcode13=5 % ^^M
108 \endlinechar=13 %
109 \catcode35=6 % #
110 \catcode39=12 % '
111 \catcode44=12 % ,
112 \catcode45=12 % -
113 \catcode46=12 % .
114 \catcode58=12 % :
115 \catcode64=11 % @
116 \catcode123=1 % {
117 \catcode125=2 % }
118 \expandafter\let\expandafter\x\csname ver@kvoptions.sty\endcsname
119 \ifx\x\relax % plain-TeX, first loading
120 \else
121 \def\empty{}%
122 \ifx\x\empty % LaTeX, first loading,
123 % variable is initialized, but \ProvidesPackage not yet seen
124 \else
125 \expandafter\ifx\csname PackageInfo\endcsname\relax
126 \def\x#1#2{%
127 \immediate\write-1{Package #1 Info: #2.}%
128 }%
129 \else
130 \def\x#1#2{\PackageInfo{#1}{#2, stopped}}%
131 \fi
132 \x{kvoptions}{The package is already loaded}%
133 \aftergroup\endinput
134 \fi
135 \fi
136 \endgroup%
```

Package identification:

```
137 \begingroup\catcode61\catcode48\catcode32=10\relax%
138 \catcode13=5 % ^^M
139 \endlinechar=13 %
140 \catcode35=6 % #
```

```

141 \catcode39=12 % '
142 \catcode40=12 % (
143 \catcode41=12 % )
144 \catcode44=12 % ,
145 \catcode45=12 % -
146 \catcode46=12 % .
147 \catcode47=12 % /
148 \catcode58=12 % :
149 \catcode64=11 % @
150 \catcode91=12 % [
151 \catcode93=12 % ]
152 \catcode123=1 % {
153 \catcode125=2 % }
154 \expandafter\ifx\csname ProvidesPackage\endcsname\relax
155   \def\x#1#2#3[#4]{\endgroup
156     \immediate\write-1{Package: #3 #4}%
157     \xdef#1{#4}%
158   }%
159 \else
160   \def\x#1#2[#3]{\endgroup
161     #2[#{#3}]%
162     \ifx#1\@undefined
163       \xdef#1{#3}%
164     \fi
165     \ifx#1\relax
166       \xdef#1{#3}%
167     \fi
168   }%
169 \fi
170 \expandafter\x\csname ver@kvoptions.sty\endcsname
171 \ProvidesPackage{kvoptions}%
172 [2022-06-15 v3.15 Key value format for package options (HO)]%

```

Catcodes

```

173 \begingroup\catcode61\catcode48\catcode32=10\relax%
174 \catcode13=5 % ^~M
175 \endlinechar=13 %
176 \catcode123=1 % {
177 \catcode125=2 % }
178 \catcode64=11 % @
179 \def\x{\endgroup
180   \expandafter\edef\csname KVO@AtEnd\endcsname{%
181     \endlinechar=\the\endlinechar\relax
182     \catcode13=\the\catcode13\relax
183     \catcode32=\the\catcode32\relax
184     \catcode35=\the\catcode35\relax
185     \catcode61=\the\catcode61\relax
186     \catcode64=\the\catcode64\relax
187     \catcode123=\the\catcode123\relax
188     \catcode125=\the\catcode125\relax
189   }%
190 }%
191 \x\catcode61\catcode48\catcode32=10\relax%
192 \catcode13=5 % ^~M
193 \endlinechar=13 %
194 \catcode35=6 % #
195 \catcode64=11 % @
196 \catcode123=1 % {
197 \catcode125=2 % }
198 \def\TMP@EnsureCode#1#2{%
199   \edef\KVO@AtEnd{%
200     \KVO@AtEnd

```

```

201   \catcode#1=\the\catcode#1\relax
202 }%
203   \catcode#1=#2\relax
204 }
205 \TMP@EnsureCode{1}{14}% ^^A (comment)
206 \TMP@EnsureCode{2}{14}% ^^A (comment)
207 \TMP@EnsureCode{33}{12}% !
208 \TMP@EnsureCode{39}{12}% '
209 \TMP@EnsureCode{40}{12}% (
210 \TMP@EnsureCode{41}{12}% )
211 \TMP@EnsureCode{42}{12}% *
212 \TMP@EnsureCode{44}{12}% ,
213 \TMP@EnsureCode{45}{12}% -
214 \TMP@EnsureCode{46}{12}% .
215 \TMP@EnsureCode{47}{12}% /
216 \TMP@EnsureCode{58}{12}% :
217 \TMP@EnsureCode{62}{12}% >
218 \TMP@EnsureCode{91}{12}% [
219 \TMP@EnsureCode{93}{12}% ]
220 \TMP@EnsureCode{94}{7}% ^ (superscript)
221 \TMP@EnsureCode{96}{12}% `
222 \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`.

```

223 \@ifundefined{define@key}{%
224   \RequirePackage{keyval}\relax
225 }{}

```

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

```

226 \RequirePackage{ltxcmds}[2010/12/02]
227 \RequirePackage{kvsetkeys}[2007/09/29]

```

Provide macros for plain T_EX.

```

228 \@ifundefined{@x@protect}{%
229   \def\@x@protect#1\fi#2#3{%
230     \fi\protect#1%
231   }%
232   \let\@typeset@protect\relax
233 }{}
234 \@ifundefined{@currname}{%
235   \def\@currname{}%
236 }{}
237 \@ifundefined{@currentx}{%
238   \def\@currentx{}%
239 }{}

```

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

```

240 \DeclareOption{debugshow}{\catcode\@ne=9 }
241 \DeclareOption{patch}{%
242   \AtEndOfPackage{%
243     \RequirePackage{kvoptions-patch}[2019/11/29]%
244   }%
245 }

```

Optionen auswerten:

```

246 \ProcessOptions\relax

```


6.3 Option declaration macros

6.3.1 `\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^AT_EX's class/package system.

```
247 \define@key{KVO}{family}{%
248   \expandafter\edef\csname KVO@family@%
249     \currname.\@current\endcsname{#1}%
250 }
251 \def\KVO@family{%
252   \ifundefined{KVO@family@\@currname.\@current}{%
253     \@currname
254   }{%
255     \csname KVO@family@\@currname.\@current\endcsname
256   }%
257 }
```

`\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.

```
258 \define@key{KVO}{prefix}{%
259   \expandafter\edef\csname KVO@prefix@%
260     \currname.\@current\endcsname{#1}%
261 }
262 \def\KVO@prefix{%
263   \ltx@ifundefined{KVO@prefix@\@currname.\@current}{%
264     \@currname @%
265   }{%
266     \csname KVO@prefix@\@currname.\@current\endcsname
267   }%
268 }

269 \define@key{KVO}{setkeys}{%
270   \expandafter\def\csname KVO@setkeys@%
271     \currname.\@current\endcsname{#1}%
272 }
```

`\KVO@setkeys`

```
273 \def\KVO@setkeys{%
274   \ltx@ifundefined{KVO@setkeys@\@currname.\@current}{%
275     \setkeys
276   }{%
277     \csname KVO@setkeys@\@currname.\@current\endcsname
278   }%
279 }
```

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

```
280 \newcommand*\SetupKeyvalOptions{%
281   \kvsetkeys{KVO}%
282 }
```

6.3.2 \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^AT_EX's `\newif` does not check for already defined macros, therefore we add this check here to prevent the user from accidentally redefining of T_EX's primitives and other macros.

```

283 \newcommand*{\DeclareBoolOption}[2][false]{%
284   \KVO@ifdefinable{if\KVO@prefix#2}{%
285     \KVO@ifdefinable{\KVO@prefix#2true}{%
286       \KVO@ifdefinable{\KVO@prefix#2false}{%
287         \csname newif\expandafter\endcsname
288         \csname if\KVO@prefix#2\endcsname
289         \@ifundefined{\KVO@prefix#2#1}{%
290           \PackageWarning{kvoptions}{%
291             Initialization of option ‘#2’ failed,\MessageBreak
292             cannot set boolean option to ‘#1’,\MessageBreak
293             use ‘true’ or ‘false’, now using ‘false’%
294           }%
295         }{%
296           \csname\KVO@prefix#2#1\endcsname
297         }%
298         \begingroup
299         \edef\x{\endgroup
300           \noexpand\define@key{\KVO@family}{#2}[true]{%
301             \noexpand\KVO@boolkey{\@currname}%
302             \ifx\@current\@clsextension
303               \noexpand\@clsextension
304             \else
305               \noexpand\@pkgextension
306             \fi
307             {\KVO@prefix}{#2}{###1}%
308           }%
309         }%
310       \x
311     }%
312   }%
313 }%
314 }

```

`\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.

```

315 \newcommand*{\DeclareComplementaryOption}[2]{%
316   \@ifundefined{if\KVO@prefix#2}{%
317     \PackageError{kvoptions}{%
318       Cannot generate option code for ‘#1’,\MessageBreak
319       parent switch ‘#2’ does not exist%
320     }{%
321       You are inside %
322       \ifx\@current\@clsextension class\else package\fi\space
323       ‘\@currname.\@current’.\MessageBreak
324       ‘\KVO@family’ is used as family %
325       for the keyval options.\MessageBreak
326       ‘\KVO@prefix’ serves as prefix %
327       for internal switch macros.\MessageBreak
328       \MessageBreak
329     }%

```

```

330   }%
331 }{%
332   \KV0@ifdefinable{\KV0@prefix#1true}{%
333     \KV0@ifdefinable{\KV0@prefix#1false}{%
334       \expandafter\let\csname\KV0@prefix#1false\expandafter\endcsname
335         \csname\KV0@prefix#2true\endcsname
336     \expandafter\let\csname\KV0@prefix#1true\expandafter\endcsname
337       \csname\KV0@prefix#2false\endcsname

```

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

```

338   \begingroup
339   \edef\x{\endgroup
340     \noexpand\define@key{\KV0@family}{#1}[true]{%
341       \noexpand\KV0@boolkey{\@currname}%
342       \ifx\@current\@clsextension
343         \noexpand\@clsextension
344       \else
345         \noexpand\@pkgextension
346       \fi
347       {\KV0@prefix}{#1}{###1}%
348     }%
349   }%
350   \x
351 }%
352 }%
353 }%
354 }

```

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

```

355 \def\KV0@ifdefinable#1{%
356   \expandafter\@ifdefinable\csname #1\endcsname
357 }

```

`\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

```

```

358 \def\KV0@boolkey#1#2#3#4#5{%
359   \edef\KV0@param{#5}%
360   \ltx@onelevel@sanitize\KV0@param
361   \ifx\KV0@param\KV0@true
362     \expandafter\@firstofone
363   \else
364     \ifx\KV0@param\KV0@false
365       \expandafter\expandafter\expandafter\@firstofone
366     \else
367       \ifx#2\@clsextension
368         \expandafter\ClassWarning
369       \else
370         \expandafter\PackageWarning
371       \fi
372     {#1}{%
373       Value ‘\KV0@param’ is not supported by\MessageBreak
374       option ‘#4’%
375     }%
376     \expandafter\expandafter\expandafter\@gobble
377   \fi
378 }

```

```

379  {%
380    ^^A\ifx#2\@clsextension
381    ^^A \expandafter\ClassInfo
382    ^^A\else
383    ^^A \expandafter\PackageInfo
384    ^^A\fi
385    ^^A{#1}{[option] #4=\KVO@param}%
386    \csname#3#4\KVO@param\endcsname
387  }%
388 }

```

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

```

389 \def\KVO@true{true}
390 \def\KVO@false{false}
391 \ltx@onelevel@sanitize\KVO@true
392 \ltx@onelevel@sanitize\KVO@false

```

6.3.3 \DeclareStringOption

\DeclareStringOption

```

393 \newcommand*{\DeclareStringOption}[2] [] {%
394   \@ifnextchar[ {%
395     \KVO@DeclareStringOption{#1}{#2}@%
396   } {%
397     \KVO@DeclareStringOption{#1}{#2}{ } []%
398   }%
399 }

```

\KVO@DeclareStringOption

```

400 \def\KVO@DeclareStringOption#1#2#3[#4] {%
401   \KVO@ifdefinable{\KVO@prefix#2}{%
402     \@namedef{\KVO@prefix#2}{#1}%
403     \begingroup
404       \ifx\#3\%
405         \toks@{%
406           \else
407             \toks@{[#4]}%
408           \fi
409         \edef\x{\endgroup
410           \noexpand\define@key{\KVO@family}{#2}\the\toks@{%
411             ^^A\begingroup
412             ^^A \toks@{###1}%
413             ^^A \ifx\@current\@clsextension
414             ^^A \noexpand\ClassInfo
415             ^^A \else
416             ^^A \noexpand\PackageInfo
417             ^^A \fi
418             ^^A {\@currname}{%
419             ^^A [option] #2={\noexpand\the\toks@}%
420             ^^A }%
421             ^^A\endgroup
422             \noexpand\def
423             \expandafter\noexpand\csname\KVO@prefix#2\endcsname{###1}%
424           }%
425         }%
426       \x
427     }%
428 }

```

6.3.4 \DeclareVoidOption

\DeclareVoidOption

```

429 \newcommand*{\DeclareVoidOption}[2]{%
430   \begingroup
431     \let\next@gobbletwo
432     \KV0@ifdefinable{\KV0@prefix#1}{%
433       \let\next@firstofone
434     }%
435   \expandafter\endgroup
436   \next{%
437     \begingroup
438     \edef\x{\endgroup
439       \noexpand\define@key{\KV0@family}{#1}[\KV0@VOID@]{%
440         \noexpand\KV0@voidkey{\@currname}%
441         \ifx\@current\@clsextension
442           \noexpand\@clsextension
443         \else
444           \noexpand\@pkgextension
445         \fi
446         {#1}%
447         {###1}%
448         \expandafter\noexpand\csname\KV0@prefix#1\endcsname
449       }%
450     }%
451     \x
452     \begingroup
453     \toks@{#2}%
454     \expandafter\endgroup
455     \expandafter\def
456     \csname\KV0@prefix#1\expandafter\endcsname
457     \expandafter{\the\toks@}%
458   }%
459 }
460 \def\KV0@VOID@{@VOID@}

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

```

```

483  ^^A{#1}{[option] #3}%
484 }

```

6.3.5 \DeclareDefaultOption

\DeclareDefaultOption

```

485 \newcommand*{\DeclareDefaultOption}{%
486   \@namedef{KV0@default@\currname.\@current}%
487 }

```

6.3.6 \DeclareLocalOptions

\DeclareLocalOptions

```

488 \newcommand*{\DeclareLocalOptions}[1]{%
489   \comma@parse{#1}\KV0@DeclareLocalOption
490 }

```

\KV0@DeclareLocalOption

```

491 \def\KV0@DeclareLocalOption#1{%
492   \expandafter\def\csname KV0@local@\KV0@family @#1\endcsname{%
493 }

```

6.4 Dynamic options

6.4.1 \DisableKeyvalOption

```

494 \SetupKeyvalOptions{%
495   family=KV0dyn,%
496   prefix=KV0dyn@%
497 }
498 \DeclareBoolOption[true]{global}
499 \DeclareComplementaryOption{local}{global}
500 \DeclareStringOption[undef]{action}
501 \let\KV0dyn@name\relax
502 \let\KV0dyn@ext\@empty
503 \define@key{KV0dyn}{class}{%
504   \def\KV0dyn@name{#1}%
505   \let\KV0dyn@ext\@clsextension
506 }
507 \define@key{KV0dyn}{package}{%
508   \def\KV0dyn@name{#1}%
509   \let\KV0dyn@ext\@pkgextension
510 }
511 \newcommand*{\DisableKeyvalOption}[3] []{%
512   \begingroup
513     \kvsetkeys{KV0dyn}{#1}%
514     \def\x{\endgroup}%
515     \@ifundefined{KV0@action@\KV0dyn@action}{%
516       \PackageError{kvoptions}{%
517         Unknown disable action %
518         '\expandafter\strip@prefix\meaning\KV0dyn@action'\MessageBreak
519         for option '#3' in keyval family '#2'%
520       } \@ehc
521     }{%
522       \csname KV0@action@\KV0dyn@action\endcsname{#2}{#3}%
523     }%
524   \x
525 }
526 \def\KV0@action@undef#1#2{%
527   \edef\x{\endgroup
528     \ifKV0dyn@global\global\fi
529     \let

```

```

530 \expandafter\noexpand\csname KV@#1@#2\endcsname
531 \relax
532 \ifKV0dyn@global\global\fi
533 \let
534 \expandafter\noexpand\csname KV@#1@#2@default\endcsname
535 \relax
536 }%
537 ^^A\PackageInfo{kvoptions}{%
538 ^^A [option] key '#2' of family '#1'\MessageBreak
539 ^^A is disabled (undef, \ifKV0dyn@global global\else local\fi)%
540 ^^A}%
541 }
542 \def\KV0@action@ignore#1#2{%
543 \edef\x{\endgroup
544 \ifKV0dyn@global\global\fi
545 \let
546 \expandafter\noexpand\csname KV@#1@#2\endcsname
547 \noexpand@gobble
548 \ifKV0dyn@global\global\fi
549 \let
550 \expandafter\noexpand\csname KV@#1@#2@default\endcsname
551 \noexpand@empty
552 }%
553 ^^A\PackageInfo{kvoptions}{%
554 ^^A [option] key '#2' of family '#1'\MessageBreak
555 ^^A is disabled (ignore, \ifKV0dyn@global global\else local\fi)%
556 ^^A}%
557 }
558 \def\KV0@action@error{%
559 \KV0@do@action{error}%
560 }
561 \def\KV0@action@warning{%
562 \KV0@do@action{warning}%
563 }
#1 error or warning
#2 family
#3 key
564 \def\KV0@do@action#1#2#3{%
565 \ifx\KV0dyn@name\relax
566 \PackageError{kvoptions}{%
567 Action type '#1' needs package/class name\MessageBreak
568 for key '#3' in family '#2'%
569 } \@ehc
570 \else
571 \edef\x{\endgroup
572 \noexpand\define@key{#2}{#3}[] {%
573 \expandafter\noexpand\csname KV0@disable@#1\endcsname
574 {\KV0dyn@name}\noexpand\KV0dyn@ext{#3}%
575 }%
576 \ifKV0dyn@global
577 \global\let
578 \expandafter\noexpand\csname KV@#2@#3\endcsname
579 \expandafter\noexpand\csname KV@#2@#3\endcsname
580 \global\let
581 \expandafter\noexpand\csname KV@#2@#3@default\endcsname
582 \expandafter\noexpand\csname KV@#2@#3@default\endcsname
583 \fi
584 }%
585 ^^A\ifx\KV0dyn@ext\@clsextension
586 ^^A \expandafter\ClassInfo
587 ^^A\else
588 ^^A \expandafter\PackageInfo

```

```

589    ^^A\fi
590    ^^A{\KVOdyn@name}{%
591    ^^A [option] key ‘#3’ of family ‘#2’\MessageBreak
592    ^^A is disabled (#1, \ifKVOdyn@global global\else local\fi)%
593    ^^A}%
594 \fi
595 }
596 \def\KVO@disable@error#1#2#3{%
597 \ifx#2\@clsextension
598 \expandafter\ClassError
599 \else
600 \expandafter\PackageError
601 \fi
602 {#1}{%
603 Option ‘#3’ is given too late,\MessageBreak
604 now the option is ignored%
605 }\@ehc
606 }
607 \def\KVO@disable@warning#1#2#3{%
608 \ifx#2\@clsextension
609 \expandafter\ClassWarning
610 \else
611 \expandafter\PackageWarning
612 \fi
613 {#1}{%
614 Option ‘#3’ is already consumed\MessageBreak
615 and has no effect%
616 }%
617 }

```

6.5 Change option code

6.5.1 \AddToKeyvalOption

\AddToKeyvalOption

```

618 \newcommand*\AddToKeyvalOption{%
619 \@ifstar{%
620 \begingroup
621 \edef\x{\endgroup
622 \noexpand\KVO@AddToKeyvalOption{\KVO@family}%
623 }%
624 \x
625 }%
626 \KVO@AddToKeyvalOption
627 }

```

\KVO@AddToKeyvalOption

```

628 \def\KVO@AddToKeyvalOption#1#2{%
629 \@ifundefined{KV@#1@#2}{%
630 \PackageWarning{kvoptions}{%
631 Key ‘#2’ of family ‘#1’ does not exist.\MessageBreak
632 Ignoring \string\AddToKeyvalOption
633 }%
634 \@gobble
635 }{%
636 \edef\KVO@next{%
637 \noexpand\KVO@@AddToKeyvalOption
638 \expandafter\noexpand\csname KV@#1@#2\endcsname
639 }%
640 \afterassignment\KVO@next
641 \def\KVO@temp##1%
642 }%
643 }

```


`\KVO@AddToKeyvalOption`

```
644 \def\KVO@AddToKeyvalOption#1{%
645   \begingroup
646   \toks@\expandafter{#1{##1}}%
647   \toks@\expandafter{\the\expandafter\toks@\KVO@temp{##1}}%
648   \edef\x{\endgroup
649     \noexpand\def\noexpand#1###1{\the\toks@}%
650   }%
651   \x
652 }
```

6.6 Process options

6.6.1 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.

```
653 </package>
654 <*package | patch>
```

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

```
655 \def\KVO@IfDefThen#1#2{%
656   \ifx#1\ltx@undefined
657   \else
658     \ifx#1\relax
659     \else
660       #2%
661     \fi
662   \fi
663 }
```

`\KVO@GetClassOptionsList`

```
664 \def\KVO@GetClassOptionsList{%
665   \let\KVO@classoptionslist\@classoptionslist
666   \KVO@IfDefThen\@classoptionslist{%
667     \KVO@IfDefThen\XKV@documentclass{%
668       \ifx\XKV@documentclass\ltx@empty
669       \else
670         \KVO@IfDefThen\XKV@classoptionslist{%
671           \ifx\XKV@classoptionslist\ltx@empty
672           \else
673             \let\KVO@classoptionslist\XKV@classoptionslist
674           \fi
675         }%
676       \fi
677     }%
678   }%
679 }%

680 </package | patch>
681 <*package>
```

6.6.2 \ProcessKeyvalOptions

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

```
682 \newcommand*{\ProcessKeyvalOptions}{%
683   \@ifstar{%
```

```

684 \begingroup
685 \edef\x{\endgroup
686 \noexpand\KVO@ProcessKeyvalOptions{\KVO@family}}%
687 }%
688 \x
689 }%
690 \KVO@ProcessKeyvalOptions
691 }

```

```

692 \def\KVO@ProcessKeyvalOptions#1{%
693 \let\@tempc\relax
694 \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).

```

695 \ifx\@currentx\@clsextension
696 \else
697 \KVO@GetClassOptionsList
698 \ifx\KVO@classoptionslist\relax
699 \else
700 \for\KVO@CurrentOption:=\KVO@classoptionslist\do{%
701 \ifundefined{KV@#1@\expandafter\KVO@getkey
702 \KVO@CurrentOption=\@nil}}{%
703 }{%
704 \ifundefined{KVO@local@#1@\expandafter\KVO@getkey
705 \KVO@CurrentOption=\@nil}}{%
706 \ifx\KVO@Patch Y%
707 \edef\KVO@temp{%
708 \etex@unexpanded\expandafter{%
709 \KVO@temp
710 }%
711 ,%
712 \etex@unexpanded\expandafter{%
713 \KVO@CurrentOption
714 }%
715 ,%
716 }%
717 \ltx@onelevel@sanitize\KVO@CurrentOption
718 \else
719 \edef\KVO@temp{%
720 \KVO@temp
721 ,%
722 \KVO@CurrentOption
723 ,%
724 }%
725 \fi
726 \@expandtwoargs\@removeelement{\expandafter\@remove@eq@value\KVO@CurrentOption=}
727 \@unusedoptionlist\@unusedoptionlist
728 }{}%
729 }%
730 }%
731 \fi
732 \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`.

```

733 \begingroup
734 \toks\@tw@{}%
735 \ifundefined{opt@\@currname.\@currentx}{%
736 \toks\@expandafter{\KVO@temp}}%
737 }{%
738 \toks\@expandafter\expandafter\expandafter{%

```

```

739     \csname opt@\@currname.\@current\endcsname
740   }%
741   \ifx\@current\@clsextension
742     \edef\CurrentOption{\the\toks@}%
743     \toks@\expandafter{\KV0@temp}%
744     \@for\CurrentOption:=\CurrentOption\do{%
745       \ifundefined{%
746         KV@#1@\expandafter\KV0@getkey\CurrentOption=\@nil
747       }{%

```

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

```

748     \ifundefined{KV0@default@\@currname.\@current}{%
749       \ifxKV0@Patch Y%
750         \ltx@onelevel@sanitize\CurrentOption
751       \fi
752       \ifx\@unusedoptionlist\@empty
753         \global\let\@unusedoptionlist\CurrentOption
754       \else
755         \expandafter\expandafter\expandafter\gdef
756         \expandafter\expandafter\expandafter\@unusedoptionlist
757         \expandafter\expandafter\expandafter{%
758           \expandafter\@unusedoptionlist
759           \expandafter,\CurrentOption
760         }%
761       \fi
762     }{%
763       \toks\tw@\expandafter{%
764         \the\toks@\expandafter\tw@\expandafter,\CurrentOption
765       }%
766     }%
767   }{%
768     \toks@\expandafter{%
769       \the\expandafter\toks@\expandafter,\CurrentOption
770     }%
771   }%
772 }%
773 \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.

```

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

```

```

795     \fi
796   }%
797   \edef\KVO@temp{\endgroup
798     \noexpand\KVO@calldefault{\the\toks\tw@}%
799     \noexpand\KVO@setkeys{#1}{\the\toks@}%
800   }%
801   \KVO@temp

```

Some cleanup of \ProcessOptions.

```

802   \let\CurrentOption\@empty
803   \AtEndOfPackage{\let\@unprocessedoptions\relax}%
804 }

```

6.6.3 \ProcessLocalKeyvalOptions

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

```

805 \newcommand*{\ProcessLocalKeyvalOptions}{%
806   \@ifstar{%
807     \begingroup
808     \edef\x{\endgroup
809       \noexpand\KVO@ProcessLocalKeyvalOptions{\KVO@family}%
810     }%
811     \x
812   }%
813   \KVO@ProcessLocalKeyvalOptions
814 }

815 \def\KVO@ProcessLocalKeyvalOptions#1{%
816   \let\@tempc\relax
817   \let\KVO@temp\@empty

```

Check if \ProcessLocalKeyvalOptions is called inside a package.

```

818   \ifx\@current\@pkgextension
819   \else
820     \PackageError{kvoptions}{%
821       \string\ProcessLocalKeyvalOptions\space is intended for packages only%
822     }\@ehc
823   \fi

```

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

```

824   \begingroup
825     \toks\tw@{ }%
826     \@ifundefined{opt@\@currname.\@current}{%
827       \toks@\expandafter{\KVO@temp}%
828     }{%
829       \toks@\expandafter\expandafter\expandafter{%
830         \csname opt@\@currname.\@current\endcsname
831       }%

```

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.

```

832     \@ifundefined{KVO@default@\@currname.\@current}{%
833       \toks@\expandafter\expandafter\expandafter{%
834         \expandafter\KVO@temp\the\toks@
835       }%
836     }{%
837       \edef\CurrentOption{\the\toks@}%
838       \toks@\expandafter{\KVO@temp}%
839       \@for\CurrentOption:=\CurrentOption\do{%
840         \@ifundefined{%
841           KV@#1\expandafter\KVO@getkey\CurrentOption=\@nil
842         }{%
843           \toks\tw@\expandafter{%

```

```

844         \the\toks\expandafter\tw@\expandafter,\CurrentOption
845     }%
846 }{%
847     \toks@\expandafter{%
848         \the\expandafter\toks@\expandafter,\CurrentOption
849     }%
850 }%
851 }%
852 }%
853 }%
854 \edef\KVO@temp{\endgroup
855     \noexpand\KVO@calldefault{\the\toks\tw@}%
856     \noexpand\KVO@setkeys{#1}{\the\toks@}%
857 }%
858 \KVO@temp

```

Some cleanup of \ProcessOptions.

```

859 \let\CurrentOption\@empty
860 \AtEndOfPackage{\let\@unprocessedoptions\relax}%
861 }

```

6.6.4 Helper macros

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

```

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

```

`\KVO@calldefault`

```

863 \def\KVO@calldefault#1{%
864     \begingroup
865     \def\x{#1}%
866     \expandafter\endgroup
867     \ifx\x\@empty
868     \else
869         \@for\CurrentOption:=#1\do{%
870             \ifx\CurrentOption\@empty
871             \else
872                 \expandafter\KVO@setcurrents\CurrentOption=\@nil
873                 \@nameuse{KVO@default@\@currname.\@current}%
874             \fi
875         }%
876     \fi
877 }

```

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

```

878 \def\KVO@setcurrents#1=#2\@nil{%
879     \def\CurrentOptionValue{#2}%
880     \ifx\CurrentOptionValue\@empty
881         \let\CurrentOptionKey\CurrentOption
882         \let\CurrentOptionValue\relax
883     \else
884         \edef\CurrentOptionKey{\zap@space#1 \@empty}%
885         \expandafter\KVO@setcurrentvalue\CurrentOption\@nil
886     \fi
887 }

```

`\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.

```

888 \def\KV@setcurrentvalue#1=#2\@nil{%
889     \KV@@sp@def\CurrentOptionValue{#2}%
890 }

```

6.7 plain T_EX

Disable L^AT_EX stuff.

```
891 \begingroup\expandafter\expandafter\expandafter\endgroup
892 \expandafter\ifx\csname documentclass\endcsname\relax
893   \def\ProcessKeyvalOptions{%
894     \@ifstar{}\@gobble
895   }%
896 \fi

897 \KVO@AtEnd%
898 \</package>
```

6.8 Package kvoptions-patch

```
899 \<*patch>
900 \NeedsTeXFormat{LaTeX2e}
901 \begingroup\catcode61\catcode48\catcode32=10\relax%
902   \catcode13=5 % ^~M
903   \endlinechar=13 %
904   \catcode123=1 % {
905   \catcode125=2 % }
906   \catcode64=11 % @
907   \def\x{\endgroup
908     \expandafter\edef\csname KVO@AtEnd\endcsname{%
909       \endlinechar=\the\endlinechar\relax
910       \catcode13=\the\catcode13\relax
911       \catcode32=\the\catcode32\relax
912       \catcode35=\the\catcode35\relax
913       \catcode61=\the\catcode61\relax
914       \catcode64=\the\catcode64\relax
915       \catcode123=\the\catcode123\relax
916       \catcode125=\the\catcode125\relax
917     }%
918   }%
919 \x\catcode61\catcode48\catcode32=10\relax%
920 \catcode13=5 % ^~M
921 \endlinechar=13 %
922 \catcode35=6 % #
923 \catcode64=11 % @
924 \catcode123=1 % {
925 \catcode125=2 % }
926 \def\TMP@EnsureCode#1#2{%
927   \edef\KVO@AtEnd{%
928     \KVO@AtEnd
929     \catcode#1=\the\catcode#1\relax
930   }%
931   \catcode#1=#2\relax
932 }
933 \TMP@EnsureCode{39}{12}% '
934 \TMP@EnsureCode{40}{12}% (
935 \TMP@EnsureCode{41}{12}% )
936 \TMP@EnsureCode{43}{12}% +
937 \TMP@EnsureCode{44}{12}% ,
938 \TMP@EnsureCode{45}{12}% -
939 \TMP@EnsureCode{46}{12}% .
940 \TMP@EnsureCode{47}{12}% /
941 \TMP@EnsureCode{58}{12}% :
942 \TMP@EnsureCode{60}{12}% <
943 \TMP@EnsureCode{62}{12}% >
944 \TMP@EnsureCode{91}{12}% [
945 \TMP@EnsureCode{93}{12}% ]
946 \TMP@EnsureCode{96}{12}% `
```

```

947 \TMP@EnsureCode{124}{12}% |
948 \edef\KVO@AtEnd{\KVO@AtEnd\noexpand\endinput}
949 \ProvidesPackage{kvoptions-patch}%
950 [2022-06-15 v3.15 LaTeX patch for keyval options (H0)]%

  Check for  $\varepsilon$ -TeX.
951 \begingroup\expandafter\expandafter\expandafter\endgroup
952 \expandafter\ifx\csname eTeXversion\endcsname\relax
953   \PackageWarningNoLine{kvoptions-patch}{%
954     Package loading is aborted, because e-TeX is missing%
955   }%
956 \expandafter\KVO@AtEnd
957 \fi%

  Package etexcmds for \etex@unexpanded.
958 \RequirePackage{etexcmds}[2007/09/09]
959 \ifetex@unexpanded
960 \else
961   \PackageError{kvoptions-patch}{%
962     Could not find eTeX's \stringunexpanded.\MessageBreak
963     Try adding \string\RequirePackage\string{etexcmds}\string} %
964     before \string\documentclass%
965   }\@ehd
966 \expandafter\KVO@AtEnd
967 \fi%

  Check for package xkvltxp.
968 \@ifpackageloaded{xkvltxp}{%
969   \PackageWarningNoLine{kvoptions}{%
970     Option 'patch' cannot be used together with\MessageBreak
971     package 'xkvltxp' that is already loaded.\MessageBreak
972     Therefore package loading is aborted%
973   }%
974 \KVO@AtEnd
975 }{}%

976 \def\@if@options#1#2#3{%
977   \begingroup
978     \KVO@normalize\KVO@temp{#3}%
979     \edef\x{\endgroup
980       \noexpand\@if@opti@ns{%
981         \detokenize\expandafter\expandafter\expandafter{%
982           \csname opt@#2.#1\endcsname
983         }%
984       }{%
985         \detokenize\expandafter{\KVO@temp}%
986       }%
987     }%
988   \x
989 }

990 \def\@pass@options#1#2#3{%
991   \KVO@normalize\KVO@temp{#2}%
992   \@ifundefined{opt@#3.#1}{%
993     \expandafter\gdef\csname opt@#3.#1%
994       \expandafter\endcsname\expandafter{%
995       \KVO@temp
996     }%
997   }{%
998     \expandafter\gdef\csname opt@#3.#1%
999       \expandafter\expandafter\expandafter\endcsname
1000       \expandafter\expandafter\expandafter{%
1001         \csname opt@#3.#1\expandafter\endcsname\expandafter,\KVO@temp
1002       }%
1003   }%
1004 }

```

```

1005 \def\ProcessOptions{%
1006   \let\ds@\@empty
1007   \ifundefined{opt@\@currname.\@currentx}{%
1008     \let\@curroptions\@empty
1009   }{%
1010     \expandafter\expandafter\expandafter\def
1011     \expandafter\expandafter\expandafter\@curroptions
1012     \expandafter\expandafter\expandafter{%
1013       \csname opt@\@currname.\@currentx\endcsname
1014     }%
1015   }%
1016   \ifstar\KVO@xprocess@options\KVO@process@options
1017 }

1018 \def\KVO@process@options{%
1019   \@for\CurrentOption:=\@declaredoptions\do{%
1020     \ifx\CurrentOption\@empty
1021     \else
1022       \begingroup
1023         \ifx\@currentx\@clsextension
1024           \toks@{}%
1025         \else
1026           \KVO@GetClassOptionsList
1027           \toks@\expandafter{\KVO@classoptionslist,%
1028         \fi
1029         \toks\tw@\expandafter{\@curroptions}%
1030         \edef\x{\endgroup
1031           \noexpand\in@{,\CurrentOption,}{,\the\toks@\the\toks\tw@,%
1032         }%
1033         \x
1034         \ifin@
1035           \KVO@use@option
1036           \expandafter\let\csname ds@\CurrentOption\endcsname\@empty
1037         \fi
1038       \fi
1039     }%
1040   \KVO@process@ptions
1041 }

1042 \def\KVO@xprocess@options{%
1043   \ifx\@currentx\@clsextension
1044   \else
1045     \KVO@GetClassOptionsList
1046     \@for\CurrentOption:=\KVO@classoptionslist\do{%
1047       \ifx\CurrentOption\@empty
1048       \else
1049         \KVO@in@\CurrentOption\@declaredoptions
1050         \ifin@
1051           \KVO@use@option
1052           \expandafter\let\csname ds@\CurrentOption\endcsname\@empty
1053         \fi
1054       \fi
1055     }%
1056   \fi
1057   \KVO@process@ptions
1058 }

1059 \def\KVO@in@#1#2{%
1060   \in@false
1061   \begingroup
1062   \@for\x:=#2\do{%
1063     \ifx\x#1\relax
1064       \in@true
1065     \fi
1066   }%

```



```

1067 \edef\x{\endgroup
1068 \ifin@
1069 \noexpand\in@true
1070 \fi
1071 }%
1072 \x
1073 }

1074 \def\KVO@process@ptions{%
1075 \@for\CurrentOption:=\@curroptions\do{%
1076 \ifundefined{ds@\KVO@SanitizedCurrentOption}{%
1077 \KVO@use@ption
1078 \default@ds
1079 }%
1080 \KVO@use@ption
1081 }%
1082 \@for\CurrentOption:=\@declaredoptions\do{%
1083 \expandafter\let\csname ds@\CurrentOption\endcsname\relax
1084 }%
1085 \let\CurrentOption\@empty
1086 \let\@fileswith@ptions\@fileswith@ptions
1087 \AtEndOfPackage{\let\@unprocessedoptions\relax}%
1088 }

1089 \def\KVO@use@ption{%
1090 \begingroup
1091 \edef\x{\endgroup
1092 \noexpand\@removeelement{%
1093 \detokenize\expandafter{\CurrentOption}%
1094 }{%
1095 \detokenize\expandafter{\@unusedoptionlist}%
1096 }%
1097 }%
1098 \x\@unusedoptionlist
1099 \csname ds@\KVO@SanitizedCurrentOption\endcsname
1100 }

1101 \def\OptionNotUsed{%
1102 \ifx\@current\@clsextension
1103 \xdef\@unusedoptionlist{%
1104 \ifx\@unusedoptionlist\@empty
1105 \else
1106 \detokenize\expandafter{\@unusedoptionlist,}%
1107 \fi
1108 \detokenize\expandafter{\CurrentOption}%
1109 }%
1110 \fi
1111 }

Variant of \ExecuteOptions that better protects \CurrentOption.
1112 \def\CurrentOption@SaveLevel{0}
1113 \def\ExecuteOptions{%
1114 \expandafter\KVO@ExecuteOptions
1115 \csname CurrentOption@CurrentOption@SaveLevel\endcsname
1116 }
1117 \def\KVO@ExecuteOptions#1#2{%
1118 \let#1\CurrentOption
1119 \edef\CurrentOption@SaveLevel{%
1120 \the\numexpr\CurrentOption@SaveLevel+1%
1121 }%
1122 \@for\CurrentOption:=#2\do{%
1123 \csname ds@\CurrentOption\endcsname
1124 }%
1125 \edef\CurrentOption@SaveLevel{%
1126 \the\numexpr\CurrentOption@SaveLevel-1%

```

```

1127 }%
1128 \let\CurrentOption#1%
1129 }
1130 \def\KVO@fileswith@ptions#1[#2]#3[#4]{%
1131 \ifx#1\@clsextension
1132 \ifx\@classoptionslist\relax
1133 \KVO@normalize\KVO@temp{#2}%
1134 \expandafter\gdef\expandafter\@classoptionslist\expandafter{%
1135 \KVO@temp
1136 }%
1137 \def\reserved@a{%
1138 \KVO@onefilewithoptions{#3}[#{2}][#{4}]#1%
1139 \@documentclasshook
1140 }%
1141 \else
1142 \def\reserved@a{%
1143 \KVO@onefilewithoptions{#3}[#{2}][#{4}]#1%
1144 }%
1145 \fi
1146 \else
1147 \begingroup
1148 \let\KVO@temp\relax
1149 \let\KVO@onefilewithoptions\relax
1150 \let\@pkgextension\relax
1151 \def\reserved@b##1,{%
1152 \ifx\@nil##1\relax
1153 \else
1154 \ifx\relax##1\relax
1155 \else
1156 \KVO@onefilewithoptions{##1}[{\KVO@temp}][#{4}]%
1157 \@pkgextension
1158 \fi
1159 \expandafter\reserved@b
1160 \fi
1161 }%
1162 \edef\reserved@a{\zap@space#3 \@empty}%
1163 \edef\reserved@a{\expandafter\reserved@b\reserved@a,\@nil,}%
1164 \toks@{#2}%
1165 \def\KVO@temp{\the\toks@}%
1166 \edef\reserved@a{\endgroup \reserved@a}%
1167 \fi
1168 \reserved@a
1169 }
1170 \def\KVO@onefilewithoptions#1[#2][#3]#4{%
1171 \@pushfilename
1172 \xdef\@currname{#1}%
1173 \global\let\@current#4%
1174 \expandafter\let\csname\@currname.\@current-h@k\endcsname\@empty
1175 \let\CurrentOption\@empty
1176 \@reset@ptions
1177 \makeatletter
1178 \def\reserved@a{%
1179 \@ifl@aded\@current{#1}{%
1180 \@if@ptions\@current{#1}{#2}{%
1181 }{%
1182 \begingroup
1183 \@ifundefined{opt@#1.\@current}{%
1184 \def\x{%
1185 }{%
1186 \edef\x{%
1187 \expandafter\expandafter\expandafter\strip@prefix
1188 \expandafter\meaning\csname opt@#1.\@current\endcsname

```

```

1189     }%
1190 }%
1191 \def\y{#2}%
1192 \edef\y{\expandafter\strip@prefix\meaning\y}%
1193 \@latex@error{Option clash for \@cls@pkg\space #1}{%
1194     The package #1 has already been loaded %
1195     with options:\MessageBreak
1196     \space\space[\x]\MessageBreak
1197     There has now been an attempt to load it %
1198     with options\MessageBreak
1199     \space\space[\y]\MessageBreak
1200     Adding the global options:\MessageBreak
1201     \space\space
1202     \x,\y\MessageBreak
1203     to your \noexpand\documentclass declaration may fix this.%
1204     \MessageBreak
1205     Try typing \space <return> \space to proceed.%
1206 }%
1207 \endgroup
1208 }%
1209 }{%
1210 \@pass@options\@currentx{#2}{#1}%
1211 \global\expandafter
1212 \let\csname ver@\@currname.\@currentx\endcsname\@empty
1213 \InputIfFileExists
1214   {\@currname.\@currentx}%
1215   {}%
1216   {\@missingfileerror\@currname\@currentx}%
1217 \let\@unprocessedoptions\@unprocessedoptions
1218 \csname\@currname.\@currentx-h@k\endcsname
1219 \expandafter\let\csname\@currname.\@currentx-h@k\endcsname
1220   \@undefined
1221 \@unprocessedoptions
1222 }%
1223 \@ifl@ter\@currentx{#1}{#3}{%
1224 }{%
1225 \@latex@warning@no@line{%
1226     You have requested,\on@line, %
1227     version\MessageBreak
1228     #3' of \@cls@pkg\space #1,\MessageBreak
1229     but only version\MessageBreak
1230     '\csname ver@#1.\@currentx\endcsname'\MessageBreak
1231     is available%
1232 }%
1233 }%
1234 \ifx\@currentx\@clsextension\let\LoadClass\@twoloadclasserror\fi
1235 \@popfilename
1236 \@reset@ptions
1237 }%
1238 \reserved@a
1239 }

1240 \def\@unknownoptionerror{%
1241 \@latex@error{%
1242     Unknown option '\KVO@SanitizedCurrentOption' %
1243     for \@cls@pkg\space'\@currname'%
1244 }{%
1245     The option '\KVO@SanitizedCurrentOption' was not declared in %
1246     \@cls@pkg\space'\@currname', perhaps you\MessageBreak
1247     misspelled its name. %
1248     Try typing \space <return> %
1249     \space to proceed.%
1250 }%

```

```

1251 }
1252 \def\@@unprocessedoptions{%
1253   \ifx\@current\@pkgextension
1254     \ifundefined{opt@\@currname.\@current}{%
1255       \let\@curroptions\@empty
1256     }{%
1257       \expandafter\let\expandafter\@curroptions
1258       \csname opt@\@currname.\@current\endcsname
1259     }%
1260     \for\CurrentOption:=\@curroptions\do{%
1261       \ifx\CurrentOption\@empty\else\@unknownoptionerror\fi
1262     }%
1263   \fi
1264 }
1265 \def\KVO@SanitizedCurrentOption{%
1266   \expandafter\strip@prefix\meaning\CurrentOption
1267 }

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

```

```

1311 \expandafter\KVO@removelastspace\expandafter=%
1312 % "<value><spaceopt>= \@nil"
1313 \else
1314 \expandafter\KVO@checkfirstA
1315 \fi
1316 }
1317 \def\KVO@checkfirstA#1 #2\@nil{%
1318 \KVO@ifempty{#2}{%
1319 \KVO@removelastspace=#1 \@nil
1320 }{%
1321 \KVO@ifempty{#1}{%
1322 \KVO@removelastspace=#2\@nil
1323 }{%
1324 \KVO@removelastspace=#1 #2\@nil
1325 }%
1326 }%
1327 }
1328 \def\KVO@removelastspace#1 = \@nil|#2\KVO@nil{%
1329 \KVO@ifempty{#2}{%
1330 \edef\KVO@result{%
1331 \etex@unexpanded\expandafter{\KVO@result}%
1332 \etex@unexpanded\expandafter{\KVO@removegarbage#1\KVO@nil}%
1333 }%
1334 }{%
1335 \edef\KVO@result{%
1336 \etex@unexpanded\expandafter{\KVO@result}%
1337 \etex@unexpanded{#1}%
1338 }%
1339 }%
1340 }
1341 \def\KVO@removegarbage#1= \@nil#2\KVO@nil{#1}%
Arguments #1 and #2 are macros.
1342 \def\KVO@removeelement#1#2{%
1343 \begingroup
1344 \toks@={}%
1345 \@for\x:=#2\do{%
1346 \ifx\x\@empty
1347 \else
1348 \ifx\x#1\relax
1349 \else
1350 \edef\t{\the\toks@}%
1351 \ifx\t\@empty
1352 \else
1353 \toks@\expandafter{\the\toks@,}%
1354 \fi
1355 \toks@\expandafter{\the\expandafter\toks@\x}%
1356 \fi
1357 \fi
1358 }%
1359 \edef\x{\endgroup
1360 \def\noexpand#2{\the\toks@}%
1361 }%
1362 \x
1363 }
1364 \let\@@fileswith@pti@ns\KVO@fileswith@pti@ns
1365 \ifx\@fileswith@pti@ns\@badrequireerror
1366 \else
1367 \let\@fileswith@pti@ns\KVO@fileswith@pti@ns
1368 \fi

```

\KVO@Patch

```
1369 \let\KVO@Patch=Y
```

```
1370 \KV0@AtEnd%
1371 </patch>
```

7 Installation

7.1 Download

Package. This package is available on CTAN¹:

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

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

7.2 Package installation

Unpacking. The `.dtx` file is a self-extracting `docstrip` archive. The files are extracted by running the `.dtx` through plain \TeX :

```
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/kvoptions/kvoptions.sty
kvoptions-patch.sty    → tex/latex/kvoptions/kvoptions-patch.sty
kvoptions.pdf          → doc/latex/kvoptions/kvoptions.pdf
example-mycolorsetup.sty → doc/latex/kvoptions/example-mycolorsetup.sty
kvoptions.dtx          → source/latex/kvoptions/kvoptions.dtx
```

If you have a `docstrip.cfg` that configures and enables `docstrip`'s TDS installing feature, then some files can already be in the right place, see the documentation of `docstrip`.

7.3 Refresh file name databases

If your \TeX distribution (\TeX Live, MiK \TeX , ...) relies on file name databases, you must refresh these. For example, \TeX Live users run `texhash` or `mktexlsr`.

7.4 Some details for the interested

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.

¹[CTAN:pkg/kvoptions](#)

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

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

An example follows how to generate the documentation with pdfL^AT_EX:

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

8 References

- [1] A guide to key-value methods, Joseph Wright, second draft for TUGBoat, 2009-03-17. <https://www.texdev.net/uploads/2009/03/keyval.pdf>
- [2] Package `ifthen`, David Carlisle, 2001/05/26. CTAN:pkg/[ifthen](#)
- [3] Package `helvet`, Sebastian Rahtz, Walter Schmidt, 2004/01/26. CTAN:pkg/[psfonts](#)
- [4] Package `hyperref`, Sebastian Rahtz, Heiko Oberdiek, 2006/02/12. CTAN:pkg/[hyperref](#)
- [5] Package `keyval`, David Carlisle, 1999/03/16. CTAN:pkg/[keyval](#)
- [6] Package `multicol`, Frank Mittelbach, 2004/02/14. CTAN:pkg/[multicol](#)
- [7] Package `tabularx`, David Carlisle, 1999/01/07. CTAN:pkg/[tabularx](#)
- [8] Package `tracefnt`, Frank Mittelbach, Rainer Schöpf, 1997/05/29. CTAN:pkg/[latex-base](#)
- [9] Package `xkeyval`, Hendri Adriaens, 2005/05/07. CTAN:pkg/[xkeyval](#)
- [10] The L^AT_EX3 Project, *L^AT_EX 2_ε for class and package writers*, 2003/12/09. CTAN:pkg/[clsguide](#)

9 History

[0000/00/00 v0.0]

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

[2004/02/22 v1.0]

- The first version was never published. It also has offered a patch to get rid of L^AT_EX’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^AT_EX. 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_EX 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.

[2019/11/29 v3.13]

- Documentation updates.

[2020-10-07 v3.14]

- `kvoptions-patch` is not compatible with a `LATEX 2020-10-01` or newer and so it will abort loading if it detects it. This fixes github issue #5.

[2022-06-15 v3.15]

- Remove only the key from the `\@unusedoptionlist`

10 Index

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

Symbols	
<code>\@fileswith@ptions</code>	1086, 1364
<code>\@unprocessedoptions</code> . . .	1217, 1252
<code>\@SetupDriver</code>	41, 43
<code>\@badrequireerror</code>	1365
<code>\@classoptionslist</code> 665, 666, 1132, 1134	
<code>\@cls@pkg</code>	1193, 1228, 1243, 1246
<code>\@clsextension</code>	
.	302, 303, 322, 342, 343, 367, 380, 413, 441, 442, 468, 478, 505, 585, 597, 608, 695, 741, 1023, 1043, 1102, 1131, 1234
<code>\@current</code> 238, 249, 252, 255, 260, 263, 266, 271, 274, 277, 302, 322, 323, 342, 413, 441, 486, 695, 735, 739, 741, 748, 774, 818, 826, 830, 832, 873, 1007, 1013, 1023, 1043, 1102, 1173, 1174, 1179, 1180, 1183, 1188, 1210, 1212, 1214, 1216, 1218, 1219, 1223, 1230, 1234, 1253, 1254, 1258	
<code>\@currname</code>	57, 235, 249, 252, 253, 255, 260, 263, 264, 266, 271, 274, 277, 301, 323, 341, 418, 440, 486, 735, 739, 748, 774, 826, 830, 832, 873, 1007, 1013, 1172, 1174, 1212, 1214, 1216, 1218, 1219, 1243, 1246, 1254, 1258
<code>\@curroptions</code>	1008, 1011, 1029, 1075, 1255, 1257, 1260
<code>\@declaredoptions</code>	1019, 1049, 1082
<code>\@documentclasshook</code>	1139
<code>\@ehc</code>	329, 520, 569, 605, 822
<code>\@ehd</code>	965
<code>\@empty</code>	88, 89, 502, 551, 694, 752, 802, 817, 859, 867, 870, 880, 884, 1006, 1008, 1020, 1036, 1047, 1052, 1085, 1104, 1162, 1174, 1175, 1212, 1255, 1261, 1269, 1289, 1290, 1303, 1346, 1351
<code>\@expandtwoargs</code>	726
<code>\@fileswith@ptions</code>	1086, 1365, 1367
<code>\@firstofone</code>	362, 365, 433
<code>\@firstoftwo</code>	1281
<code>\@for</code>	700, 744, 781, 839, 869, 1019, 1046, 1062, 1075, 1082, 1122, 1260, 1345
<code>\@gobble</code>	376, 547, 634, 894
<code>\@gobbletwo</code>	431
<code>\@if@ptions</code>	980
<code>\@if@ptions</code>	976, 1180
<code>\@ifdefinable</code>	356
<code>\@ifl@aded</code>	1179
<code>\@ifl@t@r</code>	99
<code>\@ifl@ter</code>	1223
<code>\@ifnextchar</code>	394
<code>\@ifpackageloaded</code>	968
<code>\@ifstar</code>	619, 683, 806, 894, 1016
<code>\@ifundefined</code>	
.	223, 228, 234, 237, 252, 289, 316, 515, 629, 701, 704, 735, 745, 748, 774, 782, 826, 832, 840, 992, 1007, 1076, 1183, 1254
<code>\@latex@error</code>	1193, 1241
<code>\@latex@warning@no@line</code>	1225
<code>\@let@token</code>	1305, 1309
<code>\@missingfileerror</code>	1216
<code>\@namedef</code>	402, 486
<code>\@nameuse</code>	873
<code>\@ne</code>	240
<code>\@nil</code>	702, 705, 726, 746, 783, 841, 862, 872, 878, 885, 888, 1152, 1163, 1270, 1273, 1275, 1277, 1286, 1305, 1312, 1317, 1319, 1322, 1324, 1328, 1341
<code>\@pass@ptions</code>	990, 1210
<code>\@pkgextension</code>	305, 345, 444, 509, 818, 1150, 1157, 1253
<code>\@popfilename</code>	1235
<code>\@pushfilename</code>	1171
<code>\@remove@eq@value</code>	726
<code>\@removeelement</code>	726, 1092
<code>\@reset@ptions</code>	1176, 1236
<code>\@secondoftwo</code>	1283
<code>\@tempc</code>	693, 816
<code>\@twoloadclasserror</code>	1234
<code>\@typeset@protect</code>	232
<code>\@undefined</code>	162, 1220
<code>\@unknownoptionerror</code>	68, 1240, 1261
<code>\@unprocessedoptions</code>	
.	803, 860, 1087, 1217, 1221
<code>\@unusedoptionlist</code>	
.	727, 752, 753, 756, 758, 1095, 1098, 1103, 1104, 1106
<code>\@x@protect</code>	229
<code>\@</code>	404, 1280
A	
<code>\AddToKeyvalOption</code>	8, <u>618</u> , 632
<code>\afterassignment</code>	640
<code>\aftergroup</code>	133
<code>\AtEndOfPackage</code>	242, 803, 860, 1087
C	
<code>\catcode</code>	106, 107, 109, 110, 111, 112, 113, 114, 115, 116, 117, 137, 138, 140, 141, 142, 143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 173, 174, 176, 177, 178, 182, 183, 184, 185, 186, 187, 188, 191, 192, 194, 195,

196, 197, 201, 203, 240, 901, 902, 904, 905, 906, 910, 911, 912, 913, 914, 915, 916, 919, 920, 922, 923, 924, 925, 929, 931	288, 296, 334, 335, 336, 337, 356, 386, 423, 448, 456, 492, 522, 530, 534, 546, 550, 573, 578, 579, 581, 582, 638, 739, 830, 892, 908, 952, 982, 994, 999, 1001, 1013, 1036, 1052, 1083, 1099, 1115, 1123, 1174, 1188, 1212, 1218, 1219, 1230, 1258
\ClassError 598	\endinput 103, 133, 222, 948
\ClassInfo 381, 414, 479, 586	\endlinechar 108, 139, 175, 181, 193, 903, 909, 921
\ClassWarning 368, 469, 609	\etex@unexpanded 708, 712, 1295, 1302, 1331, 1332, 1336, 1337
\comma@parse 489	\ExecuteOptions 1113
\csname 118, 125, 154, 170, 180, 248, 255, 259, 266, 270, 277, 287, 288, 296, 334, 335, 336, 337, 356, 386, 423, 448, 456, 492, 522, 530, 534, 546, 550, 573, 578, 579, 581, 582, 638, 739, 830, 892, 908, 952, 982, 993, 998, 1001, 1013, 1036, 1052, 1083, 1099, 1115, 1123, 1174, 1188, 1212, 1218, 1219, 1230, 1258	F
\CurrentOption 35, 37, 38, 41, 58, 62, 64, 462, 742, 744, 746, 750, 753, 759, 764, 769, 779, 781, 783, 786, 790, 802, 837, 839, 841, 844, 848, 859, 869, 870, 872, 881, 885, 1019, 1020, 1031, 1036, 1046, 1047, 1049, 1052, 1075, 1082, 1083, 1085, 1093, 1108, 1118, 1122, 1123, 1128, 1175, 1260, 1261, 1266	\fmtversion 99, 102
\CurrentOption@SaveLevel 1112, 1115, 1119, 1120, 1125, 1126	\futurelet 1305
\CurrentOptionKey 881, 884	G
\CurrentOptionValue 56, 879, 880, 882, 889	\gdef 755, 993, 998, 1134
D	I
\DeclareBoolOption . 5, 20, 22, 283, 498	\ifetex@unexpanded 959
\DeclareComplementaryOption 6, 315, 499	\IfFormatAtLeastTF 99, 100, 103
\DeclareDefaultOption 6, 55, 485	\ifin@ 1034, 1050, 1068
\DeclareLocalOption 7	\ifKV0dyn@global 528, 532, 539, 544, 548, 555, 576, 592
\DeclareLocalOptions 488	\ifMCS@print 79
\DeclareOption 240, 241	\ifpdf 25
\DeclareStringOption 5, 26, 28, 50, 52, 393, 500	\ifx 56, 88, 119, 122, 125, 154, 162, 165, 302, 322, 342, 361, 364, 367, 380, 404, 413, 441, 466, 468, 478, 565, 585, 597, 608, 656, 658, 668, 671, 695, 698, 706, 741, 749, 752, 818, 867, 870, 880, 892, 952, 1020, 1023, 1043, 1047, 1063, 1102, 1104, 1131, 1132, 1152, 1154, 1234, 1253, 1261, 1280, 1290, 1309, 1346, 1348, 1351, 1365
\DeclareVoidOption . . 6, 32, 33, 34, 429	\immediate 127, 156
\default@ds 1078	\in@ 1031
\define@key 247, 258, 269, 300, 340, 410, 439, 503, 507, 572	\in@false 1060
\detokenize 981, 985, 1093, 1095, 1106, 1108, 1280	\in@true 1064, 1069
\DisableKeyvalOption 7, 511	\InputIfFileExists 1213
\do 700, 744, 781, 839, 869, 1019, 1046, 1062, 1075, 1082, 1122, 1260, 1345	K
\documentclass 964, 1203	\KV@@sp@def 889
\ds@ 1006	\KV@setcurrentvalue 888
E	\KV0@@AddToKeyvalOption . . . 637, 644
\emph 48, 87, 91, 93	\KV0@action@error 558
\empty 121, 122	\KV0@action@ignore 542
\endscname 118, 125, 154, 170, 180, 249, 255, 260, 266, 271, 277, 287,	\KV0@action@undef 526
	\KV0@action@warning 561
	\KV0@AddToKeyvalOption 622, 626, 628
	\KV0@AtEnd 199, 200, 222, 897, 927, 928, 948, 956, 966, 974, 1370
	\KV0@boolkey 301, 341, 358
	\KV0@calldefault 798, 855, 863
	\KV0@checkfirstA 1314, 1317
	\KV0@checkfirsttok 1305, 1308
	\KV0@checkkv 1275, 1286

<code>\KVO@classoptionslist</code>	<code>\KVO@voidkey</code>
. . . 665, 673, 698, 700, 1027, 1046	440, 461
<code>\KVO@CurrentOption</code>	<code>\KVO@x</code>
. 700, 702, 705, 713, 717, 722, 726	1289, 1290, 1295
<code>\KVO@DeclareLocalOption</code> . . . 489, 491	<code>\KVO@xprocess@ptions</code> 1016, 1042
<code>\KVO@DeclareStringOption</code> 395, 397, 400	<code>\KVOdyn@action</code>
<code>\KVO@disable@error</code>	515, 518, 522
596	<code>\KVOdyn@ext</code>
<code>\KVO@disable@warning</code>	502, 505, 509, 574, 585
607	<code>\KVOdyn@name</code> 501, 504, 508, 565, 574, 590
<code>\KVO@do@action</code>	<code>\kvsetkeys</code>
559, 562, 564	281, 513
<code>\KVO@ExecuteOptions</code>	
1114, 1117	L
<code>\KVO@false</code>	<code>\LoadClass</code>
364, 389	1234
<code>\KVO@family</code>	<code>\ltx@empty</code>
247, 300, 324,	668, 671
340, 410, 439, 492, 622, 686, 809	<code>\ltx@ifUndefined</code>
<code>\KVO@fileswith@ptions</code> 1130, 1364, 1367	274
<code>\KVO@GetClassOptionsList</code>	<code>\ltx@ifundefined</code>
664, 697, 1026, 1045	263
<code>\KVO@getkey</code> 701, 704, 746, 783, 841, 862	<code>\ltx@onelevel@sanitize</code>
<code>\KVO@ifdefinable</code>	360, 391, 392, 717, 750
284,	<code>\ltx@undefined</code>
285, 286, 332, 333, 355, 401, 432	656
<code>\KVO@ifDefThen</code>	
655, 666, 667, 670	M
<code>\KVO@ifempty</code>	<code>\makeatletter</code>
1277, 1279, 1287, 1318, 1321, 1329	1177
<code>\KVO@in@</code>	<code>\MCS@driver</code>
1049, 1059	84
<code>\KVO@next</code>	<code>\MCS@emph</code>
636, 640	88, 94
<code>\KVO@nil</code>	<code>\meaning</code>
1305, 1328, 1332, 1341	518, 1188, 1192, 1266
<code>\KVO@normalize</code>	<code>\MessageBreak</code>
978, 991, 1133, 1268	58, 101, 102, 291, 292, 318,
<code>\KVO@onefilewithoptions</code>	323, 325, 327, 328, 373, 474,
1138, 1143, 1149, 1156, 1170	518, 538, 554, 567, 591, 603,
<code>\KVO@param</code>	614, 631, 962, 970, 971, 1195,
359, 360, 361, 364, 373, 385, 386	1196, 1198, 1199, 1200, 1202,
<code>\KVO@Patch</code>	1204, 1227, 1228, 1229, 1230, 1246
706, 749, 1369	
<code>\KVO@prefix</code>	N
258, 284, 285, 286,	<code>\NeedsTeXFormat</code>
288, 289, 296, 307, 316, 326,	4, 900
332, 333, 334, 335, 336, 337,	<code>\newcommand</code> 40, 43, 280, 283, 315, 393,
347, 401, 402, 423, 432, 448, 456	429, 485, 488, 511, 618, 682, 805
<code>\KVO@process@ptions</code> . 1040, 1057, 1074	<code>\next</code>
<code>\KVO@process@ptions</code>	431, 433, 436
1016, 1018	<code>\numexpr</code>
<code>\KVO@ProcessKeyvalOptions</code>	1120, 1126
686, 690, 692	
<code>\KVO@ProcessLocalKeyvalOptions</code>	O
809, 813, 815	<code>\on@line</code>
<code>\KVO@removeelement</code>	1226
1342	<code>\OptionNotUsed</code>
<code>\KVO@removegarbage</code>	1101
1332, 1341	
<code>\KVO@removelastspace</code>	P
1311, 1319, 1322, 1324, 1328	<code>\PackageError</code>
<code>\KVO@result</code> 1269, 1271, 1294, 1295,	317, 516, 566, 600, 820, 961
1301, 1302, 1330, 1331, 1335, 1336	<code>\PackageInfo</code>
<code>\KVO@SanitizedCurrentOption</code>	130, 383, 416, 481, 537, 553, 588
1076, 1099, 1242, 1245, 1265	<code>\PackageWarning</code>
<code>\KVO@setcurrents</code>	100, 290, 370, 471, 611, 630
872, 878	<code>\PackageWarningNoLine</code> . . 57, 953, 969
<code>\KVO@setcurrentvalue</code>	<code>\PassOptionsToPackage</code>
885, 888	63, 80
<code>\KVO@setkeys</code>	<code>\ProcessKeyvalOptions</code> 3, 74, 682, 893
273, 799, 856	<code>\ProcessLocalKeyvalOptions</code> 4, 805, 821
<code>\KVO@splitcomma</code>	<code>\ProcessOptions</code>
1270, 1273, 1277	246, 1005
<code>\KVO@temp</code> . 641, 647, 694, 707, 709,	<code>\protect</code>
719, 720, 736, 743, 776, 780,	230
797, 801, 817, 827, 834, 838,	<code>\providecommand</code>
854, 858, 978, 985, 991, 995,	99
1001, 1133, 1135, 1148, 1156, 1165	<code>\ProvidesPackage</code>
<code>\KVO@true</code>	5, 123, 171, 949
361, 389	
<code>\KVO@use@ption</code>	R
1035, 1051, 1077, 1080, 1089	<code>\renewcommand</code>
<code>\KVO@VOID@</code>	93
439, 460, 466	<code>\RequirePackage</code>
	7,
	8, 84, 224, 226, 227, 243, 958, 963
	<code>\reserved@a</code>
	1137, 1142,
	1162, 1163, 1166, 1168, 1178, 1238

<code>\reserved@b</code>	1151, 1159, 1163	743, 768, 769, 775, 776, 779, 780, 789, 790, 799, 827, 829, 833, 834, 837, 838, 847, 848, 856, 1024, 1027, 1031, 1164, 1165, 1344, 1350, 1353, 1355, 1360
S		
<code>\setkeys</code>	44, 275	
<code>\SetupDriver</code>	32, 33, 34, 35, 38, 40	
<code>\SetupKeyvalOptions</code>	4, 13, 280, 494	<code>\tw@</code> 734, 763, 764, 785, 786, 798, 825, 843, 844, 855, 1029, 1031
<code>\space</code>	322, 821, 1193, 1196, 1199, 1201, 1205, 1228, 1243, 1246, 1248, 1249	
<code>\strip@prefix</code>	518, 1187, 1192, 1266	
T		
<code>\t</code>	1350, 1351	
<code>\textcolor</code>	94	
<code>\the</code>	181, 182, 183, 184, 185, 186, 187, 188, 201, 410, 419, 457, 647, 649, 742, 764, 769, 776, 779, 786, 790, 798, 799, 834, 837, 844, 848, 855, 856, 909, 910, 911, 912, 913, 914, 915, 916, 929, 1031, 1120, 1126, 1165, 1350, 1353, 1355, 1360	
<code>\TMP@EnsureCode</code>	198, 205, 206, 207, 208, 209, 210, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220, 221, 926, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947	
<code>\toks</code>	734, 763, 764, 785, 786, 798, 825, 843, 844, 855, 1029, 1031	
<code>\toks@</code>	405, 407, 410, 412, 419, 453, 457, 646, 647, 649, 736, 738, 742,	
U		
<code>\unexpanded</code>	962	
W		
<code>\write</code>	127, 156	
X		
<code>\x</code>	118, 119, 122, 126, 130, 132, 155, 160, 170, 179, 191, 299, 310, 339, 350, 409, 426, 438, 451, 464, 466, 514, 524, 527, 543, 571, 621, 624, 648, 651, 685, 688, 808, 811, 865, 867, 907, 919, 979, 988, 1030, 1033, 1062, 1063, 1067, 1072, 1091, 1098, 1184, 1186, 1196, 1202, 1345, 1346, 1348, 1355, 1359, 1362	
<code>\XKV@classoptionslist</code>	670, 671, 673	
<code>\XKV@documentclass</code>	667, 668	
Y		
<code>\y</code>	1191, 1192, 1199, 1202	
Z		
<code>\zap@space</code>	884, 1162, 1289, 1303	