\textbf{expkvopt}  
parse class and package options with \texttt{expkv}  

Jonathan P. Spratte\textsuperscript{*}  
2020-07-04 \texttt{vo.1a}

\section*{Abstract}
\texttt{expkvopt} provides option parsing for classes and packages in \LaTeX\, based on \texttt{expkv}. Global and local options are parsed individually by different commands. The stylised name is \texttt{expkvopt} but the files use \texttt{expkv-opt}, this is due to CTAN-rules which don't allow \texttt{|} in package names since that is the pipe symbol in \texttt{*nix} shells.

\section*{Contents}
\begin{itemize}
\item[1] Documentation \hfill 2
  \begin{itemize}
  \item[1.1] Macros \hfill 2
  \item[1.2] Example \hfill 3
  \item[1.3] Bugs \hfill 5
  \item[1.4] License \hfill 5
  \end{itemize}
\item[2] Implementation \hfill 6
  \begin{itemize}
  \item[2.1] Loop \hfill 6
  \item[2.2] Tests \hfill 6
  \item[2.3] Key handlers \hfill 7
  \item[2.4] Processing list elements \hfill 9
  \item[2.5] List variable helpers \hfill 10
  \item[2.6] Errors \hfill 11
  \item[2.7] User Interface \hfill 11
  \end{itemize}
\end{itemize}

\section*{Index}
\begin{itemize}
\item[\textsuperscript{*}] jspratte@yahoo.de
\end{itemize}
1 Documentation

The \texttt{expkv} family provides at its core a \(\langle \text{key} \rangle = \langle \text{value} \rangle\) parser and additionally packages, one to conveniently define new keys (\texttt{expkv\def}) and another to build expandable \(\langle \text{key} \rangle = \langle \text{value} \rangle\) taking control sequences (\texttt{expkv\cs}). Still missing from the mix was a solution to parse \LaTeX{} class and package options, a gap that’s hereby filled with \texttt{expkv\opt}.

\texttt{expkv\opt} shouldn’t place any restrictions on the keys, but note that parts of \LaTeX{} can break if the \(\langle \text{key} \rangle = \langle \text{value} \rangle\) list contains braces. This includes the global options list depending on which class you’re using. Also keep in mind that every value provided should be save from an \texttt{edef} expansion, as the space stripping code of \LaTeX{} options (which is applied before \texttt{expkv\opt} takes control) uses such an expansion.

The package can be loaded with

\begin{verbatim}
\usepackage{expkv-opt}
\end{verbatim}

Unlike the other packages in the \texttt{expkv} family, \texttt{expkv\opt} is only provided as a \LaTeX{} package.

Before reading this documentation you should read \texttt{expkv}\def’s documentation and might want to also read the documentation of \texttt{expkv\def}.

1.1 Macros

\texttt{expkv\opt}’s behaviour if it encounters a defined or an undefined \(\langle \text{key} \rangle\) depends on which list is being parsed and whether the current file is a class or not. Of course in every case a defined \(\langle \text{key} \rangle\)’s callback will be invoked but an additional action might be executed. For this reason the rule set of every macro will be given below the short description which list it will parse.

During each of the processing macros the current list element (not separated in \(\langle \text{key} \rangle\) and \(\langle \text{value} \rangle\) but as a whole) is stored within the macro \texttt{\CurrentOption}.

\begin{verbatim}
\ekvoProcessLocalOptions{\langle \set \rangle}
\end{verbatim}

This parses the options which are directly passed to the current class or package for an \texttt{expkv} \(\langle \set \rangle\).

\begin{description}
  \item[Class: defined] \texttt{nothing}
  \item[undefined] add the key to the list of unused global options (if the local option list matches the option list of the main class)
\end{description}

\begin{verbatim}
\ekvoProcessGlobalOptions{\langle \set \rangle}
\end{verbatim}

In \LaTeX{} the options given to \texttt{\documentclass} are global options. This macro processes the global options for an \texttt{expkv} \(\langle \set \rangle\).

\begin{description}
  \item[Class: defined] remove the option from the list of unused global options
  \item[undefined] \texttt{nothing}
\end{description}
Package: defined  remove the option from the list of unused global options
undefined  nothing

\ekvoProcessUnusedGlobalOptions  \ekvoProcessUnusedGlobalOptions{(set)}

If you want to, instead of parsing all global options, you can parse only those global options which weren’t yet used by another package or class.

Class: defined  remove the option from the list of unused global options
undefined  nothing

Package: defined  remove the option from the list of unused global options
undefined  nothing

\ekvoProcessOptionsList  \ekvoProcessOptionsList{list}{(set)}

Process the \textit{(key)}=\textit{(value)} list stored in the macro \textit{list}.

Class: defined  nothing
undefined  nothing

Package: defined  nothing
undefined  nothing

\ekvoUseUnknownHandlers  \ekvoUseUnknownHandlers{(cs}_1\rangle{(cs}_2\rangle)

With this macro you can change the action \texttt{expkvopt} executes if it encounters an undefined \textit{(key)} for the next (and only the next) list processing macro. The macro \texttt{(cs}_1\rangle will be called if an undefined \textit{(key)} without a \textit{(value)} is encountered and get one argument, being the \textit{(key)}. Analogous the macro \texttt{(cs}_2\rangle will be called if an undefined \textit{(key)} with a \textit{(value)} was specified. It will get two arguments, the first being the \textit{(key)} and the second the \textit{(value)}.

\ekvoVersion  \ekvoDate

These two macros store the version and date of the package.

1.2 Example

Let’s say we want to create a package that changes the way footnotes are displayed in \LaTeX. For this it will essentially just redefine \texttt{\thefootnote} and we’ll call this package \texttt{ex-footnote}. First we report back which package we are:

\texttt{\ProvidesPackage{ex-footnote}{2020–02–02 v1 change footnotes}}

Next we’ll need to provide the options we want the package to have.

\texttt{\RequirePackage{color}}
\texttt{\RequirePackage{expkv-opt}} % also loads expkv
\texttt{\ekvdef{ex-footnote}{color}{\def\exfn@color{#1}}}\texttt{\ekvdef{ex-footnote}{format}{\def\exfn@format{#1}}}
We can provide initial values just by defining the two macros storing the value.
\newcommand{\exfn@color}{}
\newcommand{\exfn@format}{arabic}

Next we need to process the options given to the package. The package should only obey
options directly passed to it, so we’re only using \texttt{\ekvoProcessLocalOptions}:
\texttt{\ekvoProcessLocalOptions{\texttt{ex–footnote}}}

Now everything that’s still missing is actually changing the way footnotes appear:
\texttt{\renewcommand{\thefootnote}{%}
  \texttt{\texttt{\texttt{i}}f x \texttt{\exfn@color}@empty}
  \texttt{\csname \exfn@format@endcsname\footnote} %
  \texttt{\texttt{\texttt{e}}lse}
  \texttt{\texttt{\texttt{\texttt{t}}e\texttt{\texttt{x}}\texttt{\texttt{c}}olor [\texttt{\exfn@color}@\csname \texttt{\exfn@format@endcsname\footnote} ]} %
  \texttt{\texttt{\texttt{f}}i}%
\texttt{}}

So the complete code of the package would look like this:
\texttt{\ProvidesPackage{ex–footnote}[2020–02–02 v1 change footnotes]}
\texttt{\RequirePackage{color}}
\texttt{\RequirePackage{expkv–opt} % also loads expkv}
\texttt{\ekvdef{ex–footnote}[color]{\def{\exfn@color}[@1]}}
\texttt{\ekvdef{ex–footnote}[format]{\def{\exfn@format}@[1]}}
\texttt{\newcommand{\exfn@color}{} } 
\texttt{\newcommand{\exfn@format}{arabic}}
\texttt{\ekvoProcessLocalOptions{\texttt{ex–footnote}}} 
\texttt{\renewcommand{\thefootnote}{%}
  \texttt{\texttt{\texttt{i}}f x \texttt{\exfn@color}@\texttt{empty}}
  \texttt{\csname \exfn@format@endcsname\footnote} %
  \texttt{\texttt{\texttt{e}}lse}
  \texttt{\texttt{\texttt{\texttt{t}}e\texttt{\texttt{c}}olor [\texttt{\exfn@color}@\csname \texttt{\exfn@format@endcsname\footnote} ]} %
  \texttt{\texttt{\texttt{f}}i}%
\texttt{}}

And it could be used with one of the following lines:
\texttt{\usepackage{ex–footnote}}
\texttt{\usepackage[format=fnsymbol]{ex–footnote}}
\texttt{\usepackage[color=green]{ex–footnote}}
\texttt{\usepackage[color=red/format=roman]{ex–footnote}}
1.3 Bugs
If you happen to find bugs, it’d be great if you let me know. Just write me an email (see the front page) or submit a bug report on GitHub: https://github.com/Skillmon/tex_expkv-opt

1.4 License
Copyright © 2020 Jonathan P. Spratte
This work may be distributed and/or modified under the conditions of the \LaTeX{} Project Public License (LPPL), either version 1.3c of this license or (at your option) any later version. The latest version of this license is in the file:
http://www.latex-project.org/lppl.txt
This work is “maintained” (as per LPPL maintenance status) by Jonathan P. Spratte.
2 Implementation

Start the package with the typical \LaTeX standards.

\ekvoVersion
\ekvoDate

Store the packages version and date in two macros.

\newcommand*{\ekvoVersion}{0.1a}
\newcommand*{\ekvoDate}{2020-07-04}

(End definition for \ekvoVersion and \ekvoDate. These functions are documented on page 3.)

And we report who we are and what we need.

\ProvidesPackage{expkv-opt}
\RequirePackage{expkv}

2.1 Loop

We’ll need some loop which can iterate over a comma separated list. The loop is very basic and only works for commas of category 12. First we insert the delimiters for the actual loop.

\protected\long\def{\ekvo@CurrentOption@loop}{1#2\%}
{\ekvo@CurrentOption@loop@2\ekvo@mark1,\ekvo@stop,\ekvo@tail}

The actual loop checks whether the final element has been read and if so ends the loop. Else blank elements are ignored, \CurrentOption is set and the macro which parses the list elements called. Then call the next iteration.

\long\def{\ekvo@CurrentOption@loop@}{1#2,\%}
{\ekv@gobble@from@mark@to@stop2\ekvo@end@loop\ekvo@stop
\ekvo@ifblank{#2}{}}
{\edef{\CurrentOption}{\unexpanded\expandafter{\@gobble#2}}%\#1(#2)%}
}{\ekvo@CurrentOption@loop@1\ekvo@mark}

(End definition for \ekvo@CurrentOption@loop, \ekvo@CurrentOption@loop@, and \ekvo@end@loop.)

2.2 Tests

We’ll need branching $\texttt{\textbackslash ifx}$ tests so that user input containing unbalanced \TeX ifs doesn’t break (at least not because of us, everything else is the fault of \LaTeX).

\def{\ekvo@ifx@TF}{1#2{\ekv@fi@firstoftwo{\texttt{\textbackslash fi}}}2}\ekvo@ifx@F{1#2{\ekv@fi@secondoftwo{\texttt{\textbackslash fi}}}1}\ekv@fi@firstofone}

(End definition for \ekvo@ifx@TF and \ekvo@ifx@F.)
This test checks whether the \texttt{⟨set⟩} is defined. If it is we store it in \texttt{ekvo@setname} and set \texttt{ekvo@name} to a short cut to get the \texttt{⟨key⟩}’s callback name. Next we execute the code in \#2, if the \texttt{⟨set⟩} isn’t defined \#2 is gobbled.

\begin{verbatim}
\protected\def\ekvo@do@with@set#1#2\{
  \ekvifdefined\set{#1}\{
    \expandafter\let\expandafter\ekvo@name\csname\ekv@undefined@set{#1}\endcsname\def\ekvo@setname{#1}\}
  \ekvo@err@undefined@set{#1}\}
\end{verbatim}

\textit{\(\text{End definition for \texttt{ekvo@do@with@set, ekvo@name, and ekvo@setname.}\)}\}

\section{Key handlers}

\texttt{expkvopt} uses handlers specifying what happens if a parsed \texttt{⟨key⟩} is defined or undefined.

The case for undefined keys in a local list of a package is easy, just throw appropriate errors.

\begin{verbatim}
\protected\long\def\ekvo@handle@undefined@k@pkg#1\{
  \ekv@ifdefined{\ekvo@name{#1}}\{
    \ekvo@err@value@required{#1}\}
  \ekvo@err@undefined@key{#1}\}
\end{verbatim}

\begin{verbatim}
\protected\long\def\ekvo@handle@undefined@kv@pkg#1#2\{
  \ekv@ifdefined{\ekvo@name{#1}N}\{
    \ekvo@err@value@forbidden{#1}\}
  \ekvo@err@undefined@key{#1}\}
\end{verbatim}

\textit{\(\text{End definition for \texttt{ekvo@handle@undefined@k@pkg and ekvo@handle@undefined@kv@pkg.}\)}\}

\texttt{\ekvo@addto@unused@one} \texttt{\ekvo@addto@unused@two} \texttt{\ekvo@rmfrom@unused@one} \texttt{\ekvo@rmfrom@unused@two} These macros will add or remove the \texttt{\CurrentOption} to or from the list of unused global options.

\begin{verbatim}
\long\def\ekvo@addto@unused@one#1\{\ekvo@addto@list@\unusedoptionlist\}
\long\def\ekvo@addto@unused@two#1\{\ekvo@addto@list@\unusedoptionlist\}
\long\def\ekvo@rmfrom@unused@one#1\{\ekvo@rmfrom@list@\unusedoptionlist\}
\long\def\ekvo@rmfrom@unused@two#1\{\ekvo@rmfrom@list@\unusedoptionlist\}
\end{verbatim}

\textit{\(\text{End definition for \texttt{ekvo@addto@unused@one and others.}\)}\}

\texttt{\ekvo@set@handlers@local} \texttt{\ekvo@set@handlers@global} \texttt{\ekvo@set@handlers@unusedglobal} \texttt{\ekvo@set@handlers@list} These macros are boring. They just set up the handlers to respect the rules documented earlier.

\begin{verbatim}
\protected\def\ekvo@set@handlers@local\{
\ekvo@if@need@handlers\{
\end{verbatim}

\textit{\(\text{End definition for \texttt{ekvo@set@handlers@local and others.}\)}\}

7
\ifx\@currext\@clsextension
  \ifx\@classoptionslist\relax
    \let\ekvo@handle@undefined@k\@gobble
    \let\ekvo@handle@undefined@kv\@gobbletwo
  \else
    \expandafter
    \ifx\csname opt@\@currname.\@currext\endcsname\@classoptionslist
      \let\ekvo@handle@undefined@k\ekvo@addto@unused@one
      \let\ekvo@handle@undefined@kv\ekvo@addto@unused@two
    \else
      \let\ekvo@handle@undefined@k\@gobble
      \let\ekvo@handle@undefined@kv\@gobbletwo
    \fi
  \fi
\else
  \let\ekvo@handle@undefined@k\ekvo@handle@undefined@k@pkg
  \let\ekvo@handle@undefined@kv\ekvo@handle@undefined@kv@pkg
\fi
\%
\protected\def\ekvo@set@handlers@global
\{
  \unless\ifx\@unusedoptionlist\@empty
    \let\ekvo@handle@defined@k\ekvo@rmfrom@unused@one
    \let\ekvo@handle@defined@kv\ekvo@rmfrom@unused@two
  \fi
  \ekvo@if@need@handlers
    \{
      \let\ekvo@handle@undefined@k\@gobble
      \let\ekvo@handle@undefined@kv\@gobbletwo
    \}
  \}
\protected\def\ekvo@set@handlers@unusedglobal
\{
  \ekvo@if@need@handlers
    \{
      \let\ekvo@handle@undefined@k\ekvo@handle@undefined@k@pkg
      \let\ekvo@handle@undefined@kv\ekvo@handle@undefined@kv@pkg
    \}
  \}
\protected\def\ekvo@set@handlers@list
\{
  \ekvo@if@need@handlers
    \{
      \let\ekvo@handle@undefined@k\@gobble
      \let\ekvo@handle@undefined@kv\@gobbletwo
    \}
  \}
\protected\def\ekvo@set@handlers@list
\{
  \ekvo@if@need@handlers
    \{
      \let\ekvo@handle@undefined@k\@gobble
      \let\ekvo@handle@undefined@kv\@gobbletwo
    \}
  \}
\%
\numberline{8}
\ekvo@if@need@handlers
\ekvo@dont@need@handlers

If the user specifies handlers this macro will be let to \ekvo@dont@need@handlers, which
will act like \@gobble and also let it to \@firstofone afterwards.

\let\ekvo@if@need@handlers\@firstofone

\protected\long\def\ekvo@dont@need@handlers#1% {
\let\ekvo@if@need@handlers\@firstofone
}

(End definition for \ekvo@if@need@handlers and \ekvo@dont@need@handlers.)

We have to set the default for the handlers of defined keys, because they don’t
necessarily get defined before a list is parsed.
\let\ekvo@handle@defined@k\@gobble
\let\ekvo@handle@defined@kv\@gobbletwo

2.4 Processing list elements

\ekvo@process@common

All the key processing frontend macros use the same basic structure. #1 will be a simple
test, deciding whether the list will really be parsed or not, #3 will be the ⟨set⟩, and #2
will be the individual code of the frontend macro which should be executed if both the
test in #1 is true and the ⟨set⟩ is defined.

\protected\def\ekvo@process@common#1#2#3% {
#1{\ekvo@do@with@set{#3}{#2}}
}

(End definition for \ekvo@process@common.)

\ekvo@process@list

This macro only expands the list holding macro and forwards it to the loop macro.

\protected\def\ekvo@process@list#1% {
\expandafter\ekvo@CurrentOption@loop\expandafter{#1}\ekvo@parse
}

(End definition for \ekvo@process@list.)

\ekvo@parse

This macro calls internals of \ekvparse such that the code splitting at commas isn’t
executed, else this is equivalent to \ekvparse\ekvo@set@k\ekvo@set@kv(#1).

\long\def\ekvo@parse#1% {
\ekvo@eq@other#1\ekv@nil\ekv@mark\ekv@parse@eq@other@a =\ekv@mark\ekv@parse@eq@active\ekv@stop
\ekvo@set@k\ekvo@set@kv
}

(End definition for \ekvo@parse.)
These two macros check whether the key is defined and if so call the handler for defined keys and execute the key, else the handler for undefined keys is called.

\begin{verbatim}
\protected\def\ekvo@set@k#1\{% \\
  \ekvo@ifdefined{\ekvo@name{#1}N}% \\
  {\ekvo@handle@defined@k{#1} N\endcsname} \\
  \ekvo@handle@undefined@k{#1}%
\}
\protected\def\ekvo@set@kv#1#2\{% \\
  \ekvo@ifdefined{\ekvo@name{#1}}% \\
  {\ekvo@handle@defined@kv{#1}{#2} N\endcsname{#2}} \\
  \ekvo@handle@undefined@kv{#1}{#2}%
\}
\end{verbatim}

(End definition for \ekvo@set@k and \ekvo@set@kv.)

\section{List variable helpers}

\ekvo@addto@list

This macro is rather simple. If the list to which the \texttt{CurrentOption} should be added is empty we can just let the list to the \texttt{CurrentOption}. Else we have to expand the list once and the \texttt{CurrentOption} once.

\begin{verbatim}
\protected\def\ekvo@addto@list#1\{% \\
  \ekvo@ifx@TF#1\@empty \\
  {\let#1\CurrentOption} \\
  {\\edef#1\{
    \unexpanded\expandafter{#1},\CurrentOption
  \}} \\
\}
\end{verbatim}

(End definition for \ekvo@addto@list.)

\ekvo@rmfrom@list

This works by looping over every list item and comparing it to \texttt{ekvo@currropt} which stores the real \texttt{CurrentOption}. This is comparatively slow, but works for items containing braces unlike what \LaTeX{} does. We could be faster for items not containing braces, though.

\begin{verbatim}
\protected\def\ekvo@rmfrom@list#1\{% \\
  \ekvo@ifx@F#1\@empty \\
  {\\let\ekvo@tmp@list\@empty \\
    \let\ekvo@currropt\CurrentOption \\
  } \\
\}
\end{verbatim}

(End definition for \ekvo@rmfrom@list.)
170 \expandafter\ekvo@CurrentOption@loop\expandafter{#1}\ekvo@rmfrom@list@  
171 \let\CurrentOption\ekvo@curropt 
172 \let#1\ekvo@tmp@list  
173 \%  
174 }%  
175 \protected\long\def\ekvo@rmfrom@list@#1%  
176 {  
177 \ekvo@ifx@F\CurrentOption\ekvo@curropt}
178 {\ekvo@addto@list\ekvo@tmp@list}\%  
179  
2(End definition for \ekvo@rmfrom@list and \ekvo@rmfrom@list@.)

2.6 Errors

Just some macros to throw errors in the few cases an error has to be thrown.

\ekvo@err@undefined@key
\ekvo@err@value@required
\ekvo@err@value@forbidden
\ekvo@err@undefined@set

\ekvo@err@undefined@key#1%  
\ekvo@err@value@required#1%  
\ekvo@err@value@forbidden#1%  
\ekvo@err@undefined@set#1%  

(End definition for \ekvo@err@undefined@key and others.)

2.7 User Interface

The user interface macros just put together the bits and pieces.

\ekvoProcessLocalOptions

\protected\def\ekvoProcessLocalOptions
\ekvo@process@common{\ekv@ifdefined{opt@\@currname.\@currext}\@firstofone\@gobble}  
\ekvo@set@handlers@local
\expandafter\ekvo@process@list\csname opt@\@currname.\@currext\endcsname  

(End definition for \ekvoProcessLocalOptions and \ekvoProcessLocalOptions.)
(End definition for \texttt{\texttt{ekvoProcessLocalOptions}}. This function is documented on page 2.)

\texttt{ekvoProcessGlobalOptions}

\begin{verbatim}
\protected\def\ekvoProcessGlobalOptions
  {%
    \ekvo@process@common{\ekvo@ifx@F\@classoptionslist\relax}%
    {% \ekvo@set@handlers@global
      \ekvo@process@list\@classoptionslist
      \let\ekvo@handle@defined@k\@gobble
      \let\ekvo@handle@defined@kv\@gobbletwo
    }%
  }
\end{verbatim}

(End definition for \texttt{\texttt{ekvoProcessGlobalOptions}}. This function is documented on page 2.)

\texttt{ekvoProcessUnusedGlobalOptions}

\begin{verbatim}
\protected\def\ekvoProcessUnusedGlobalOptions
  {%
    \ekvo@process@common{\ekvo@ifx@F\@unusedoptionlist\@empty}%
    {% \let\ekvo@tmp@list\@unusedoptionlist
      \ekvo@set@handlers@unusedglobal
      \ekvo@process@list\@ekvo@tmp@list
      \let\ekvo@handle@defined@k\@gobble
      \let\ekvo@handle@defined@kv\@gobbletwo
    }%
  }
\end{verbatim}

(End definition for \texttt{\texttt{ekvoProcessUnusedGlobalOptions}}. This function is documented on page 3.)

\texttt{ekvoProcessOptionsList}

\begin{verbatim}
\protected\def\ekvoProcessOptionsList#1%
  {%
    \ekvo@process@common{\ekvo@ifx@F\#1\@empty}%
    {% \ekvo@set@handlers@list
      \ekvo@process@list#1%
    }%
  }
\end{verbatim}

(End definition for \texttt{\texttt{ekvoProcessOptionsList}}. This function is documented on page 3.)

\texttt{ekvoUseUnknownHandlers}

\begin{verbatim}
\protected\def\ekvoUseUnknownHandlers#1#2%
  {% \let\ekvo@handle@undefined@k#1\relax
    \let\ekvo@handle@undefined@kv#2\relax
    \let\ekvo@if@need@handlers\ekvo@dont@need@handlers
  }
\end{verbatim}
All user interface macros should be only used in the preamble.

\@onlypreamble\ekvoProcessLocalOptions
\@onlypreamble\ekvoProcessGlobalOptions
\@onlypreamble\ekvoProcessUnusedGlobalOptions
\@onlypreamble\ekvoProcessOptionsList
\@onlypreamble\ekvoUseUnknownHandlers
The italic numbers denote the pages where the corresponding entry is described, numbers underlined point to the definition, all others indicate the places where it is used.

Index

The italic numbers denote the pages where the corresponding entry is described, numbers underlined point to the definition, all others indicate the places where it is used.

C
\CurrentOption 2, 19, 155, 160, 169, 171, 177
\ekvoProcessGlobalOptions ... 2, 213, 249
\ekvoProcessLocalOptions ... 2, 202, 248
\ekvoProcessUnusedGlobalOptions ... 3, 234, 251
\ekvoUseUnknownHandlers ... 3, 242, 252
\ekvoVersion ... 2, 3, 5

T\TeX and \LaTeX\TeX\ commands:
\ekvo@addto@unused@one ... 32
\ekvo@addto@list ... 50, 51, 152, 178
\ekvo@addto@unused@one ... 50, 95, 94
\ekvo@addto@unused@two ... 50, 66, 95
\ekvo@CurrentOption@loop ... 9, 126, 170
\ekvo@CurrentOption@loop@ ... 9
\ekvo@current ... 169, 171, 177
\ekvo@do@with@set ... 27, 122
\ekvo@err@undefined@set ... 36, 180
\ekvo@err@value@forbidden ... 47, 180
\ekvo@err@value@required ... 41, 180
\ekvo@handle@defined@ok ... 81, 101, 118, 138, 219, 230
\ekvo@handle@defined@kv ... 82, 102, 119, 147, 220, 231
\ekvo@handle@undefined@ok ... 61, 66, 74, 75, 94, 109, 141, 244
\ekvo@handle@undefined@ok@pkg ... 38, 73
\ekvo@handle@undefined@kv ... 38, 74
\ekvo@ifneed@handlers ... 56, 84, 92, 107, 113, 246
\ekvo@ifx@F ... 25, 166, 177, 215, 225, 236
\ekvo@ifx@TF ... 25, 154
\ekvo@name ... 27, 40, 46, 136, 139, 145, 148
\ekvo@parse ... 126, 128
\ekvo@process@common ... 120, 204, 215, 225, 236
\ekvo@process@list ... 124, 209, 218, 229, 239
\ekvo@rmfrom@list ... 52, 53, 164
\ekvo@rmfrom@list@ ... 164
\ekvo@rmfrom@undefined@one ... 50, 81, 101
\ekvo@rmfrom@unused@two ... 50, 82, 102
\ekvo@set@handlers@global ... 54, 217
\ekvo@set@handlers@list ... 54, 238
\ekvo@set@handlers@local ... 54, 207
\ekvo@set@handlers@undefined@global ... 54, 228
\ekvo@set@set@0 ... 132, 134
\ekvo@set@kv ... 132, 134
\ekvo@set@name ... 27, 182, 187, 193
\ekvo@tail ... 11, 24
\ekvo@tmp@list ... 169, 172, 178, 227, 229

Index