

What's the correct way to define package options in expl3?

Asked 7 years, 4 months ago Modified 8 months ago Viewed 2k times

20/10/2024 09:36



For writing a package it's usually important to be able to define package options. Since the package I'm writing is in expl3 I wondered whether there's a I3ish way of defining key-value-package-options.



21

I know there's 13keys as part of expl3, but as far as I can tell, it does not evaluate options given to the package directly, but has to be called using a wrapper macro (can that be done automatically?).



Looking at the source code of <code>fontspec</code> it seems to utilize standard LaTeX2e's option processing facilities. Concluding from that I would use <code>kvoptions</code> for key-value-interfaces even for I3, although it creates I2 constructions such as <code>\newif</code> s.

On the other hand there's siunitx which uses the (poorly documented) package l3keys2e (@DavidCarlisle I do not want to reverse engineer) and the l3keys syntax for kv-option-processing. But is that a safe or recommended way? In my package I intend to use l3keys anyway.

To visualize what I want (a prototypical nonsense-MWE):

```
\documentclass{article}
 \usepackage[
     quack,% should evaluate to quack=true
     font=sffamily,% should set a font token list to sffamily
     logo=false,% should set a boolean to false
     size=3% should set an integer
 ]{mypack}
 \begin{document}
     \quackfig
 \end{document}
with mypack.sty:
 \RequirePackage{expl3,xparse}
 \ProvidesExplPackage{mypack}{2017-05-26}{v1.0}{Quack}
 \RequirePackage{graphicx}
 \bool_new:N \l__mypack_optkeys_quack_bool
 \bool_set_true:N \l__mypack_optkeys_quack_bool
 \tl_new:N \l__mypack_optkeys_font_tl
 \bool_new:N \l__mypack_optkeys_logo_bool
 \int_new:N \l__mypack_optkeys_size_int
 \int_set:Nn \l__mypack_optkeys_size_int {10}
 % Process options here
 \DeclareOption*{\PassOptionsToClass{\CurrentOption}{article}}
 \ProcessOptions\relax
```

2 sur 5 20/10/2024 09:36

```
\NewDocumentCommand{\quackfig}{}{
  \begin{figure}[htb]
  \bool_if:NT \l__mypack_optkeys_quack_bool {
      \tl_if_eq:nnT {\l__mypack_optkeys_font} {sffamily} {\sffamily}
      Quack!
  }
  \bool_if:NT \l__mypack_optkeys_logo_bool {
      \includegraphics[height=\l__mypack_optkeys_size_int cm]{example-image}
  }
  \end{figure}
}
```

QUESTION: What is the recommended way for kv-option-processing with I3 and how would mypack.sty look then?



- Could you elaborate on the lack of documentation in l3keys2e, perhaps on LaTeX-L or by
 direct mail? Joseph Wright ♦ May 26, 2017 at 7:54
- @JosephWright Currently not, I'll try my best to remember when I have got some more free time. The main point however is that for me it's pretty unclear, which options specifically are processed by \ProcessKeysPackageOptions and \ProcessKeysOptions and whether there's a way to hide some options from those commands (only to set with a wrapper). But probably that's another question. TeXnician May 26, 2017 at 7:57
- Side remark: I would avoid to declare to many package options. A setup \usepackage{myquack}\myquacksetup{...} is much more flexible and avoids option clash errors if the package is loaded more than once. (It also looks odd that you are passing an option to a class, but this perhaps only meant as example). Ulrike Fischer May 26, 2017 at 7:59
 - @UlrikeFischer Without that passing the test document would not compile for me (that's absolutely not intended in the real example). Concerning the package options I would at least like to have the flexibility (and know how to do it, as I know with I2e). Whether that will be applied is open by now. TeXnician May 26, 2017 at 8:01

1 Answer

```
Sorted by:

Reset to default

Date modified (newest first)
```

3 sur 5



Irrespective of the key processor you are using, making use of keyval options for packages has two distinct parts:

16

Defining the keys



Processing the package option list using the keyval parser



What is crucial to understand here is that keyval package options are simply keys which have been defined at the point that the option list is processed.



From 2022-06-01, the LaTeX kernel has built-in support for keyval options handling; prior to that, very similar support was provided by \lambda3keys2e. In either case, the approach uses 13keys to define options. This is done by defining the keys then processing the options

```
\providecommand \IfFormatAtLeastTF { \@ifl@t@r \fmtversion }
\ExplSyntax0n
\keys_define:nn { mypkg }
  { pkgopt .tl_set:N = \l__mypkg_pkgopt_tl }
\IfFormatAtLeastTF { 2022-06-01 }
  { \ProcessKeyOptions }
    \RequirePackage { l3keys2e }
    \ProcessKeysOptions { mypkg }
 }
\ProcessKeysOptions { mypkg } % Parses the option list
\keys_define:nn { mypkg }
  { notpkgopt .tl_set:N = \l__mypkg_notopt_tl }
```

The above will define one keyval option for the package: pkgopt, which will store the given value in the \l_mypkg_pkgopt_tl. The second key notpkgopt is defined after keyval processing so is not available to \ProcessKey(s)Options: it is therefore not a package option.

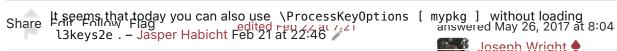
If one wishes to define all keys 'up front' but only make some available as options, the usual approach would be to use multiple paths, say mypkg / pkgopts and mypkg / otheropts.

One common use for this approach is to disable an option which is strictly load-time only. That is built-in for the kernel from 2022-06-01

```
\keys_define:nn { mypkg }
 {
   pkgopt .tl_set:N = \l__mypkg_pkgopt_tl ,
   pkgopt .usage:n. = load-only
 }
```

As noted in a comment, it is usually best to favour a setup command (\mynkqsetup) which

20/10/2024 09:36 4 sur 5



Joseph Wright ♦

1 — @JasperHabicht Updated for the newer kerenel – Joseph Wright ♦ Fel264 let 735 718 11

5 sur 5 20/10/2024 09:36