

The `l3keys2e` package*

Parsing \LaTeX 3 keyvals as \LaTeX 2 ϵ package options

The \LaTeX 3 Project†

2010/10/13

1 Key–value arguments as \LaTeX 2 ϵ class and package options

The key–value method for optional arguments is very popular, as it allows the class or package author to define a large number of options with a simple interface. The `expl3` bundle of \LaTeX 3 base code includes the module `l3keys` for defining keys, but to use these when loading \LaTeX 2 ϵ packages and classes requires extra support. That support is provided by this small package, which is intended to enable \LaTeX 2 ϵ packages to benefit from the power of the \LaTeX 3 key–value system.

1.1 Creating and using keyval options

As with any key–value input, using key–value pairs as package or class options has two parts. The first stage is to define one or more keys, using the `\keys_define:nn` function. For example, an option which simply stores a value would be created using:

```
\keys_define:nn { module } {  
  option .set:N = \l_module_variable_tl  
}
```

On its own, this will not make the key an option for the package or class containing the definition. The second stage is therefore to process the current options, searching for applicable keys.

*This file has version number 2076, last revised 2010/10/13.

†Frank Mittelbach, Denys Duchier, Chris Rowley, Rainer Schöpf, Johannes Braams, Michael Downes, David Carlisle, Alan Jeffrey, Morten Høgholm, Thomas Lotze, Javier Bezos, Will Robertson, Joseph Wright

`\ProcessKeysOptions` `\ProcessKeysOptions {\module}`

The `\ProcessKeysOptions` function is used to check the current option list against the keys defined for `{\module}`. Global (class) options and local (package) options are checked when this function is called in a package. Each option which does match a key name is then used to attempt to set the appropriate key using `\keys_set:nn`. For example, the option defined earlier would be processed by the line

```
\ProcessKeysOptions { module }
```

`\ProcessKeysPackageOptions` `\ProcessKeysPackageOptions {\module}`

This function works in a similar manner to `\ProcessKeysOptions`. When used in a \LaTeX 2_ϵ package, `\ProcessKeysPackageOptions` will not examine any class options available. In contrast, `\ProcessKeysOptions` does parse class options (in common with the \LaTeX 2_ϵ kernel function `\ProcessOptions`).

1.2 Implementation

The usual lead-off.

```
1 \*package)
2 \ProvidesExplPackage
3   {\filename}{\filedate}{\fileversion}{\filedescription}
4 \RequirePackage { xparse }
```

`\keys_latex_options_clist` A single list is used for all options, into which they are collected before processing.

```
5 \clist_new:N \keys_latex_options_clist
```

`_l_keys_process_class_bool` A flag to indicate that class options should be processed for packages.

```
6 \bool_new:N \_l_keys_process_class_bool
```

`\keys_latex_options:n` The main function calls functions to collect up the global and local options into `\keys_latex_options_clist` before calling the underlying functions to actually do the processing. So that a suitable message is produced if the option is unknown, the special `unknown` key is set if it does not already exist for the current module.

```
7 \cs_new_protected:Npn \keys_latex_options:n #1 {
8   \clist_clear:N \keys_latex_options_clist
9   \keys_latex_options_global:n {#1}
10  \keys_latex_options_local:
11  \cs_if_exist:cTF { \c_keys_root_tl {#1} / unknown .cmd:n }
12    {
```

```

13     \keys_define:nn {#1}
14     {
15         unknown .code:n =
16         {
17             \msg_kernel_error:nxxx { keyvalue } { option-unknown }
18             { \l_keys_key_tl } { \@currname }
19         }
20     }
21 }
22 \keys_set:nV {#1} \keys_latex_options_clist
23 \AtEndOfPackage { \cs_set_eq:NN \@unprocessedoptions \scan_stop: }
24 }

```

`\keys_latex_options_global:n` Global (class) options are handled differently for L^AT_EX 2_ε packages and classes. Hence this function is essentially a check on the current file type. The initial test is needed as L^AT_EX 2_ε allows variables to be equal to `\scan_stop:`, which is forbidden in L^AT_EX 3 code.

```

25 \cs_new_protected:Npn \keys_latex_options_global:n #1 {
26     \cs_if_eq:NNF \@classoptionslist \scan_stop:
27     {
28         \cs_if_eq:NNTF \@currentext \@clsextension
29         { \keys_latex_options_class:n {#1} }
30         {
31             \bool_if:NT \l_keys_process_class_bool
32             { \keys_latex_options_package:n {#1} }
33         }
34     }
35 }

```

`\keys_latex_options_class:n` For classes, each option (stripped of any content after ‘=’) is checked for existence as a key. If found, the option is added to the combined list for processing. On the other hand, unused options are stored up in `\@unusedoptionlist`.

```

36 \cs_new_protected:Npn \keys_latex_options_class:n #1 {
37     \clist_map_inline:Nn \@classoptionslist
38     {
39         \cs_if_exist:cTF
40         {
41             \c_keys_root_tl #1 /
42             \keys_latex_remove_equals:n {##1} .cmd:n
43         }
44         { \clist_put_right:Nn \keys_latex_options_clist {##1} }
45         { \clist_put_right:Nn \@unusedoptionlist {##1} }
46     }
47 }

```

`\keys_latex_options_package:n` For global options when processing a package, the tasks are slightly different from those for a class. The check is the same, but here there is nothing to do if the option is not applicable. Each valid option also needs to be removed from `\@unusedoptionlist`.

```

48 \cs_new_protected:Npn \keys_latex_options_package:n #1 {
49   \clist_map_inline:Nn \@classoptionslist
50   {
51     \cs_if_exist:cTF
52     {
53       \c_keys_root_tl #1 /
54       \keys_latex_remove_equals:n {##1} .cmd:n
55     }
56     {
57       \clist_put_right:Nn \keys_latex_options_clist {##1}
58       \clist_remove_element:Nn \@unusedoptionlist {##1}
59     }
60   }
61 }

```

`\keys_latex_options_local:` If local options are found, they are added to the processing list. L^AT_EX 2_ε stores options for each file in a macro which may or may not exist, hence the need to use `\cs_if_exist:c`.

```

62 \cs_new_protected_nopar:Npn \keys_latex_options_local: {
63   \cs_if_eq:NNF \@currentx \@clsextension
64   {
65     \cs_if_exist:cT { opt@ \@currname . \@currentx }
66     {
67       \exp_args:Nnc \clist_put_right:NV \keys_latex_options_clist
68       { opt@ \@currname . \@currentx }
69     }
70   }
71 }

```

`\keys_latex_remove_equals:n` As the name suggests, this is a simple function to remove an equals sign from the input.
`\keys_latex_remove_equals:w` This is all wrapped up in an `n` function so that there will always be a sign available.

```

72 \cs_new:Npn \keys_latex_remove_equals:n #1 {
73   \keys_latex_remove_equals:w #1 = \q_stop
74 }
75 \cs_new:Npn \keys_latex_remove_equals:w #1 = #2 \q_stop {#1}

```

`\ProcessKeysOptions` The user macro are simply wrappers around the internal process. In contrast to other similar packages, the module name is always required here.

```

76 \cs_new_protected:Npn \ProcessKeysOptions #1 {
77   \bool_set_true:N \_l_keys_process_class_bool
78   \keys_latex_options:n {#1}
79 }
80 \cs_new_protected:Npn \ProcessKeysPackageOptions #1 {
81   \bool_set_false:N \_l_keys_process_class_bool
82   \keys_latex_options:n {#1}
83 }
84 \@onlypreamble \ProcessKeysOptions
85 \@onlypreamble \ProcessKeysPackageOptions

```

One message to give.

```
86 \msg_kernel_new:nnnn { keyvalue } { option-unknown }
87 { Unknown~option~'#1'~for~package~#2. }
88 {
89   LaTeX~has~been~asked~to~set~an~option~called~'#1'\\
90   but~the~#2~package~has~not~created~an~option~with~this~name.
91 }

92 </package>
```