The l3keys2e package*
Parsing LaTeX3 keyvals as LaTeX 2ε package options

The LaTeX3 Project†

2009/08/24

1 Key–value arguments as LaTeX 2ε 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 LaTeX3 base code includes the module l3keys for defining keys, but to use these when loading LaTeX 2ε packages and classes requires extra support. That support is provided by this small package, which is intended to enable LaTeX 2ε packages to benefit from the power of the LaTeX3 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 1494, last revised 2009/08/24.

†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.

```
⟨*package⟩
\ProvidesExplPackage
  {\filename}{\filedate}{\fileversion}{\filedescription}
\RequirePackage { xparse }
```

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

```
\clist_new:N \keys_latexe_options_clist
```

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

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

\keys_latexe_options:n The main function calls functions to collect up the global and local options into `\keys_latexe_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.

```
\cs_new_protected:Npn \keys_latexe_options:n #1 {
  \clist_clear:N \keys_latexe_options_clist
  \keys_latexe_options_global:n {#1}
  \keys_latexe_options_local:
  \cs_if_exist:cTF { \c_keys_root_tl {#1} / unknown .cmd:n }
    {
```

```
      \keys_define:nn {#1}
        {
          unknown .code:n =
            {
              \msg_kernel_error:nnxx { keyvalue } { option-unknown }
                { \l_keys_key_tl } { \@currname }
            }
        }
    }
  \keys_set:nV {#1} \keys_latexe_options_clist
  \AtEndOfPackage { \cs_set_eq:NN \@unprocessedoptions \scan_stop: }
}
```

`\keys_latexe_options_global:n` Global (class) options are handled differently for LaTeX 2$_\varepsilon$ packages and classes. Hence this function is essentially a check on the current file type. The initial test is needed as LaTeX 2$_\varepsilon$ allows variables to be equal to `\scan_stop:`, which is forbidden in LaTeX3 code.

```
\cs_new_protected:Npn \keys_latexe_options_global:n #1 {
  \cs_if_eq:NNF \@classoptionslist \scan_stop:
    {
      \cs_if_eq:NNTF \@currext \@clsextension
        { \keys_latexe_options_class:n {#1} }
        {
          \bool_if:NT \_l_keys_process_class_bool
           { \keys_latexe_options_package:n {#1} }
        }
    }
}
```

`\keys_latexe_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`.

```
\cs_new_protected:Npn \keys_latexe_options_class:n #1 {
  \clist_map_inline:Nn \@classoptionslist
    {
      \cs_if_exist:cTF
        {
          \c_keys_root_tl #1 /
            \keys_latexe_remove_equals:n {##1} .cmd:n
        }
        { \clist_put_right:Nn \keys_latexe_options_clist {##1} }
        { \clist_put_right:Nn \@unusedoptionlist {##1} }
    }
}
```

`\keys_latexe_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`.

```
\cs_new_protected:Npn \keys_latexe_options_package:n #1 {
  \clist_map_inline:Nn \@classoptionslist
    {
      \cs_if_exist:cTF
        {
          \c_keys_root_tl #1 /
            \keys_latexe_remove_equals:n {##1} .cmd:n
        }
        {
          \clist_put_right:Nn \keys_latexe_options_clist {##1}
          \clist_remove_element:Nn \@unusedoptionlist {##1}
        }
    }
}
```

`\keys_latexe_options_local:` If local options are found, the are added to the processing list. LaTeX 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`.

```
\cs_new_protected_nopar:Npn \keys_latexe_options_local: {
  \cs_if_eq:NNF \@currext \@clsextension
    {
      \cs_if_exist:cT { opt@ \@currname . \@currext }
        {
          \exp_args:NNc \clist_put_right:NV \keys_latexe_options_clist
            { opt@ \@currname . \@currext }
        }
    }
}
```

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

```
\cs_new:Npn \keys_latexe_remove_equals:n #1 {
  \keys_latexe_remove_equals:w #1 = \q_stop
}
\cs_new:Npn \keys_latexe_remove_equals:w #1 = #2 \q_stop {#1}
```

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

```
\cs_new_protected:Npn \ProcessKeysOptions #1 {
  \bool_set_true:N \_l_keys_process_class_bool
  \keys_latexe_options:n {#1}
}
\cs_new_protected:Npn \ProcessKeysPackageOptions #1 {
  \bool_set_false:N \_l_keys_process_class_bool
  \keys_latexe_options:n {#1}
}
\@onlypreamble \ProcessKeysOptions
\@onlypreamble \ProcessKeysPackageOptions
```

One message to give.

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

⟨/package⟩