pgfopts — LaTeX package options with pgfkeys*

Joseph Wright†

Released 2008/06/26

Abstract

Using key–value options for packages and macros is a good way of handling large numbers of options with a clean interface. The pgfkeys package provides a very well designed system for defining and using keys, but does not make this available for handling LaTeX class and package options. The pgfopts package adds this ability to pgfkeys, in the same way that kvoptions extends the keyval package.

Contents

1 Introduction

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. A number of packages can be used to provide the key management system, most of which load or extent the parent keyval package. On its own, keyval can only be used for parsing the arguments of macros. However, a number of packages have extended the method to processing LaTeX class and package options. This processing is made available as part of two general-purpose packages xkeyval and kvoptions; both allow the author of a class or package to process key–value options given at load-time.

The pgfkeys package provides a somewhat different key–value system to keyval and derivatives. This uses a completely different model for defining and using keys, although for the end-user the result appears very similar. The pgfopts package allows keys defined with pgfkeys to be used as class or package options, in the same way that kvoptions extends keyval.

Users of pgfopts should be familiar with the general methods used by pgfkeys. These are outlined in the manual for the Ti*k*z and pgf bundle.

*This file describes version v1.0, last revised 2008/06/26.

†E-mail: joseph.wright@morningstar2.co.uk

2 Using the package

2.1 Creating options

To create package or class options for use with pgfopts, it is only necessary to define the appropriate keys. Taking as an example a package "MyOwnPackage", which uses the prefix `MOP` on internal macros, creating keys would take the form:

```
\pgfkeys{/MOP/.cd,
  keyone/.code=\wlog{Value `#1' given},
  keytwo/.store in=\MOP@store}
```

Here, `keyone` simply writes its argument to the log, while `keytwo` stores the value given in the `\MOP@store` macro.

An important point to notice is that the key names *do not* contain a space. This is because the LaTeX kernel removes spaces from options before they are passed to the class or package. Spaces can occur in the path to the key, but not in the key name itself. This restriction only applies to keys used as options.

2.2 Processing options

`\ProcessPgfOptions` Options should be processed using the `\ProcessPgfOptions` macro, which takes two forms. When invoked followed by a star, the system will use the name of the current file as the key path. Thus, for the example MyOwnPackage.sty, pgfopts would use the path `/MyOwnPackage`.

```
\ProcessPgfOptions*
```

When the key path does not match the current file name, the second form of `\ProcessPgfOptions` should be used. The full path to be searched should be given as the argument to the macro. Continuing the example, for MyOwnPackage.sty using path `/MOP`, the correct call is:

```
\ProcessPgfOptions{/MOP}
```

Notice that the *full* path is given here, including the leading "`/`"; this allows the advance user to alter the default path before calling `\ProcessPgfOptions`.

2.3 Specialised options

Users of kvoptions will be aware that there are two types of options which need specialised handling. First, there is the need to process options which should not be given a value. Using pgfkeys, this is achieved using the `.value forbidden` handler.

```
\pgfkeys{/MOP/.cd,
  keythree/.code=\wlog{keythree set},
  keythree/.value forbidden}
```

The second type of specialist option is the unknown option. Some packages accept arbitrary values in the options, which can then be used to load configuration files, *etc*. Once again, this is handled using correctly-defined keys.

```
\pgfkeys{/MOP/.cd,
  .unknown/.code=\wlog{%
    The unknown key `\pgfkeyscurrentname' was given with
    value `\pgfkeyscurrentvalue'}}
```

The power of the pgfkeys system is demonstrated here: no additional macros are needed to handle the specialised option types.

3 Implementation

The code here is based heavily on kvoptions, which has a similar aim to pgfopts, but works with keyval and derived packages.

`\pgfopts@id` Macro naming follows the scheme used by pgfkeys: the full package name is the prefix.

```
\NeedsTeXFormat{LaTeX2e}
\newcommand*{\pgfopts@id}{}
\def\pgfopts@id$#1: #2.#3 #4 #5-#6-#7 #8 #9${%
    #5/#6/#7\space v1.0\space}
\ProvidesPackage{pgfopts}
  [\pgfopts@id $Id: pgfopts.dtx 28 2008-06-26 11:59:34Z joseph $
   LaTeX package options with pgfkeys]
```

The pgfkeys package is needed, unsurprisingly.

```
\RequirePackage{pgfkeys}
```

`\pgfopts@catcodes` A few category codes are set correctly, to prevent odd accidents.

```
\edef\pgfopts@catcodes{%
  \catcode\string`\string ` \the\catcode\string`\`\relax
  \catcode\string`\string = \the\catcode\string`\=\relax
  \catcode\string`\string : \the\catcode\string`\:\relax
  \catcode\string`\string , \the\catcode\string`\,\relax
  \catcode\string`\string / \the\catcode\string`\/\relax
  \catcode\string`\string . \the\catcode\string`\.\relax}
\catcode\string`\` 12\relax
\catcode`\= 12\relax
\@makeother{\:}
\@makeother{\,}
\@makeother{\/}
\@makeother{\.}
```

`\ProcessPgfOptions` The user macro for processing options may have a star. In that case, `\@currname` is used for setting keys.

```
\newcommand*{\ProcessPgfOptions}{%
  \@ifstar
    {\begingroup
       \edef\@tempa{%
         \endgroup
         \noexpand\pgfopts@ProcessOptions
           {/\@currname}}%
     \@tempa}%
    {\pgfopts@ProcessOptions}}
\@onlypreamble\ProcessPgfOptions
```

\pgfopts@temp A private storage area is needed by the next macro.

```
\newcommand*{\pgfopts@temp}{}
```

\pgfopts@ProcessOptions `#1` : path

The business end of the package. This is a modified copy of the macro `\KVO@ProcessOptions` from kvoptions. Some simplifications are made due to the differences in key system, and as patching the kernel is ignored.

```
\newcommand*{\pgfopts@ProcessOptions}[1]{%
  \def\pgfopts@temp{,}%
  \ifx\@currext\@clsextension\else
    \ifx\@classoptionslist\relax\else
```

\pgfopts@CurrentOption For packages, the list of global (class) options needs to be checked. The assumption is made that only *explicitly* declared keys will be accepted as options here. The strange argument to `\pgfkeysifdefined` is needed as checking for the key itself only works if it has already been used.

```
      \@for\pgfopts@CurrentOption:=\@classoptionslist\do{%
        \pgfkeysifdefined{#1/\expandafter\pgfopts@sepkeyval%
          \pgfopts@CurrentOption=\@nil/.@cmd}
          {\edef\pgfopts@temp{%
             \pgfopts@temp\pgfopts@CurrentOption,}%
           \@expandtwoargs\@removeelement\pgfopts@CurrentOption
             \@unusedoptionlist\@unusedoptionlist}%
          {}}%
    \fi
  \fi
```

Now the local options have to be processed. If there are none, then the global list is simply copied. Otherwise, the local options are first added to the "to do" list stored in `\toks@`.

```
  \begingroup
    \@ifundefined{opt@\@currname.\@currext}
      {\toks@\expandafter{\pgfopts@temp}}
      {\toks@\expandafter\expandafter\expandafter{%
         \csname opt@\@currname.\@currext\endcsname}%
```

There is work to do if processing options of a class; unknown options from the global list have to be completely ignored.

```
       \ifx\@currext\@clsextension
         \edef\@tempa{\the\toks@}%
         \toks@\expandafter{\pgfopts@temp}%
         \@for\CurrentOption:=\@tempa\do{%
           \pgfkeysifdefined{#1/\expandafter%
             \pgfopts@sepkeyval\CurrentOption=\@nil/.@cmd}
             {\toks@\expandafter{\the\expandafter\toks@
                \expandafter,\CurrentOption}}
             {\ifx\@empty\@unusedoptionlist\@empty
                \global\let\@unusedoptionlist\CurrentOption
              \else
                \expandafter\expandafter\expandafter\gdef
                \expandafter\expandafter\expandafter
                \@unusedoptionlist\expandafter\expandafter
                \expandafter{%
```

```
                \expandafter\@unusedoptionlist
                \expandafter,\CurrentOption}%
            \fi}}%
```

For packages, the two lists are simply combined.

```
    \else
      \toks@\expandafter\expandafter\expandafter{%
          \expandafter\pgfopts@temp\the\toks@}%
    \fi}%
```

With everything done, the keys are sent to pgfkeys for processing. The default key system in pgfkeys is very different from that in kvoptions, and so the system here is much less complex.

```
  \edef\pgfopts@temp{\endgroup
    \noexpand\pgfkeys{#1/.cd,\the\toks@}}%
\pgfopts@temp
\let\CurrentOption\@empty
\AtEndOfPackage{\let\@unprocessedoptions\relax}}
```

`\pgfopts@sepkeyval` A very simply macro to separate a key from a potential value.

```
\newcommand*{\pgfopts@sepkeyval}{}
\def\pgfopts@sepkeyval#1=#2\@nil{#1}
```

Tidying up.

```
\pgfopts@catcodes
```

4 Change History

v1.0

General: First public release 1

5 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; numbers in roman refer to the code lines where the entry is used.