pgfopts — LaTeX package options with pgfkeys*

Joseph Wright†

Released 2014/07/10

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

*This file describes version v2.1a, last revised 2014/07/10.

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

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.

2 Installation

The package is supplied in `dtx` format and as a pre-extracted zip file, `pgfopts.tds.zip`. The later is most convenient for most users: simply unzip this in your local texmf directory and run `texhash` to update the database of file locations. If you want to unpack the `dtx` yourself, running `tex pgfopts.dtx` will extract the package whereas `latex pgfopts.dtx` will extract it and also typeset the documentation.

Typesetting the documentation requires a number of packages in addition to those needed to use the package. This is mainly because of the number of demonstration items included in the text. To compile the documentation without error, you will need the packages:

- csquotes
- helvet
- hypdoc
- listings
- lmodern
- mathpazo
- microtype

3 Using the package

3.1 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.

3.2 Processing options

\ProcessPgfOptions

The \ProcessPgfOptions macro is used to process package or class options using pgfkeys. When used in a package, it will also examine the available class options, and use any which match declared package options. \ProcessPgfOptions requires the pgfkeys key ⟨*path*⟩ to search for options. Thus for the example of MyOwnPackage given in the previous section, the appropriate call would be

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

Alternatively, \ProcessPgfOptions can be given with a star:

```
\ProcessPgfOptions*
```

The macro will then use the current file name as the path. This will be the name of the current class or package, as appropriate.

\ProcessPgfPackageOptions

As a complement to \ProcessPgfOptions, the macro \ProcessPgfPackageOptions is also provided. As the name indicates, this is intended for use in packages. It does *not* examine the list of global class options, processing only the list of options given for the class itself. If used in a class, the behaviour will be identical to \ProcessPgfOptions.

4 Implementation

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

```
⟨*package⟩
\ProvidesPackage{pgfopts}
  [2014/07/10 v2.1a LaTeX package options with pgfkeys]
```

The only package requires is pgfkeys itself.

```
\RequirePackage{pgfkeys}
```

\ifpgfopts@process@class

The processing of options can apply to those which are given for a class, or can be limited to those of the package only.

```
\newif\ifpgfopts@process@class
```

\pgfopts@options@clist

A comma-separated list of options to process.

```
\newcommand*\pgfopts@options@clist{}
```

\pgfopts@options@execute

A storage macro which specifies the final list of options to actually execute.

```
\newcommand*\pgfopts@options@execute{}
```

\pgfopts@key@path
\pgfopts@process@options

The main processing macro first clears the list of options to deal with, and stores the key path to use. Global and local options are then added to the "to do" list before executing the options.

```
\newcommand*\pgfopts@key@path{}
\newcommand\pgfopts@process@options[1]{%
  \def\pgfopts@options@clist{}%
  \def\pgfopts@options@execute{}%
  \def\pgfopts@key@path{#1/}%
  \ifx\@currext\@clsextension\else
    \expandafter\pgfopts@check@class@options
  \fi
  \pgfopts@process@local@options
  \pgfopts@options@execute
  \let\CurrentOption\@empty
  \AtEndOfPackage{\let\@unprocessedoptions\relax}%
}
```

\pgfopts@current@option
\pgfopts@check@class@options
fopts@check@class@options@aux

First, a check to see if class options should be processed, and if so if there are any. If so, then each option needs to be examined. Any =⟨*value*⟩ is removed, and a test is made to see if the resulting ⟨*key*⟩ was defined for the current ⟨*path*⟩. If so, the option is added to the list for processing later, and it is removed from the list of unused options. The syntax for the later process is rather complex, but LaTeX2e's function to achieve that is somewhat awkward.

```
\newcommand*\pgfopts@current@option{}
\newcommand*\pgfopts@check@class@options{%
  \ifpgfopts@process@class
    \ifx\@classoptionslist\relax\else
      \expandafter\expandafter\expandafter
        \pgfopts@check@class@options@aux
    \fi
  \fi
}
\newcommand*\pgfopts@check@class@options@aux{%
  \@for\pgfopts@current@option:=\@classoptionslist\do
    {%
      \pgfkeysifdefined
        {%
          \pgfopts@key@path
          \pgfopts@get@key@name\pgfopts@current@option
          /.@cmd%
        }%
        {%
          \pgfopts@list@add\pgfopts@options@clist\pgfopts@current@option
          \@expandtwoargs\@removeelement\pgfopts@current@option
            \@unusedoptionlist\@unusedoptionlist
        }%
        {}%
    }%
}
```

\pgfopts@local@options
pgfopts@process@local@options
s@process@local@options@aux@i
@process@local@options@aux@ii
s@process@local@options@class
ocess@local@options@class@aux
process@local@options@package

The first step in processing local options is to see if any exist. This is done inside a group to avoid polluting the hash table. If options are found, these are transferred into a storage macro and the auxiliary function is called. There is

then a fork depending on whether a package or class is being processed. Once the list is finalised, the execution macro is set up appropriately.

```
\newcommand*\pgfopts@local@options{}
\newcommand*\pgfopts@process@local@options{%
  \begingroup
    \@ifundefined{opt@\@currname.\@currext}%
      {\endgroup}%
      {%
        \toks@\expandafter\expandafter\expandafter
          {\csname opt@\@currname.\@currext\endcsname}%
        \expandafter\endgroup
        \expandafter\def\expandafter\pgfopts@local@options
          \expandafter{\the\toks@}%
        \pgfopts@process@local@options@aux@i
      }%
}
\newcommand*\pgfopts@process@local@options@aux@i{%
  \ifx\@currext\@clsextension
    \expandafter\pgfopts@process@local@options@class
  \else
    \expandafter\pgfopts@process@local@options@package
  \fi
  \ifx\pgfopts@options@clist\@empty\else
    \expandafter\pgfopts@process@local@options@aux@ii
  \fi
}
\newcommand*\pgfopts@process@local@options@aux@ii{%
  \begingroup
    \toks@\expandafter{\pgfopts@options@clist}%
    \edef\pgfopts@options@execute
      {%
        \noexpand\pgfkeys
          {%
            \pgfopts@key@path .cd,%
            \the\toks@
          }%
      }%
  \expandafter\endgroup
  \expandafter\def\expandafter\pgfopts@options@execute
    \expandafter{\pgfopts@options@execute}%
}
```

Options given for a class may not be applicable to the class itself, and so they have to be checked. First, there is a simple test for an unknown key handler: if it exists then there is no need to look at each option separately.

```
\newcommand*\pgfopts@process@local@options@class
  {%
    \pgfkeysifdefined{\pgfopts@key@path .unknown/.@cmd}%
      {\pgfopts@list@add\pgfopts@options@clist\@classoptionslist}
      {\pgfopts@process@local@options@class@aux}%
  }
\newcommand*\pgfopts@process@local@options@class@aux{%
  \@for\pgfopts@current@option:=\pgfopts@local@options\do{%
    \pgfkeysifdefined
```

```
      {%
        \pgfopts@key@path
        \pgfopts@get@key@name\pgfopts@current@option
        /.@cmd%
      }%
      {\pgfopts@list@add\pgfopts@options@clist\pgfopts@current@option}%
      {\pgfopts@list@add\@unusedoptionlist\pgfopts@current@option}%
  }%
}
```

For packages, the local options are simply added to the list already set up from the global values.

```
\newcommand*\pgfopts@process@local@options@package{%
  \pgfopts@list@add\pgfopts@options@clist\pgfopts@local@options
}
```

`\pgfopts@get@key@name` `\pgfopts@get@key@name@aux`

A simple function to leave only the key ⟨*name*⟩ in the input stream, whether there is a ⟨*value*⟩ or not. This needs to work with options which do not contain an equals sign at all. It is designed to work with a stored value.

```
\newcommand\pgfopts@get@key@name[1]{%
  \expandafter\pgfopts@get@key@name@aux#1=\@nil
}
\def\pgfopts@get@key@name@aux#1=#2\@nil{#1}
```

`\pgfopts@list@add@a@toks` `\pgfopts@list@add@b@toks` `\pgfopts@list@add@temp` `\pgfopts@list@add`

This function takes `#1` as the name of a comma-separated list and `#2` as the name of a macro containing content to add to the list. After the appropriate checks it adds the content of `#2` to the right hand end of variable `#1`.

```
\newtoks\pgfopts@list@add@a@toks
\newtoks\pgfopts@list@add@b@toks
\newcommand*\pgfopts@list@add@temp{}
\newcommand\pgfopts@list@add[2]{%
  \pgfopts@list@add@a@toks\expandafter{#2}%
  \def\pgfopts@list@add@temp{#2}%
  \pgfopts@list@add@b@toks\expandafter{#1}%
  \ifx\pgfopts@options@clist\@empty
    \edef#1{\the\pgfopts@list@add@a@toks}%
  \else
    \ifx\pgfopts@list@add@temp\@empty\else
      \edef#1%
        {\the\pgfopts@list@add@b@toks,\the\pgfopts@list@add@a@toks}%
    \fi
  \fi
}
```

`\ProcessPgfOptions` `\ProcessPgfPackageOptions` `\pgfopts@star@check`

The two user functions set the flag for class option processing. A shared internal function then checks for a star, and either adds the current name or looks to pick one up from the input stream.

```
\newcommand*\ProcessPgfOptions{%
  \pgfopts@process@classtrue
  \pgfopts@star@check
}
\newcommand*\ProcessPgfPackageOptions{%
  \pgfopts@process@classfalse
  \pgfopts@star@check
```

```
}
\newcommand*\pgfopts@star@check{%
  \@ifstar
    {%
      \begingroup
      \edef\@tempa
        {%
          \endgroup
          \noexpand\pgfopts@process@options{/\@currname}%
        }%
      \@tempa
    }%
    {\pgfopts@process@options}%
}
\@onlypreamble\ProcessPgfOptions
\@onlypreamble\ProcessPgfPackageOptions
⟨/package⟩
```

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.

R

T