pgfopts — LaTeX package options with pgfkeys[*]

Joseph Wright[†]

Released 2011/06/02

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 v2.1, last revised 2011/06/02.
[†]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 Change History

v1.0
  General: First public release 1
v2.0
  General: Complete internal rewrite 1
    New `\ProcessPgfPackageOptions` macro 3
v2.1
  General: Allow for `.unknown` handler for class options 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.