

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	1
2	Installation	2
3	Using the package	2
3.1	Creating options	2
3.2	Processing options	3
4	Implementation	3
5	Index	7

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 `Tikz` 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.

```

1 <*package>
2 \ProvidesPackage{pgfopts}
3 [2014/07/10 v2.1a LaTeX package options with pgfkeys]

```

The only package requires is `pgfkeys` itself.

```
4 \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.

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

`\pgfopts@options@clist` A comma-separated list of options to process.

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

`\pgfopts@options@execute` A storage macro which specifies the final list of options to actually execute.

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

`\pgfopts@key@path` 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.

```

8 \newcommand*\pgfopts@key@path{}
9 \newcommand\pgfopts@process@options[1]{%
10 \def\pgfopts@options@clist{}%
11 \def\pgfopts@options@execute{}%
12 \def\pgfopts@key@path{#1/}%
13 \ifx\@currentx\@clsextension\else
14 \expandafter\pgfopts@check@class@options
15 \fi
16 \pgfopts@process@local@options
17 \pgfopts@options@execute
18 \let\CurrentOption\@empty
19 \AtEndOfPackage{\let\@unprocessedoptions\relax}%
20 }

```

`\pgfopts@current@option` 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 $=\langle value \rangle$ is removed, and a test is made to see if the resulting $\langle key \rangle$ was defined for the current $\langle path \rangle$. 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.

```

21 \newcommand*\pgfopts@current@option{}
22 \newcommand*\pgfopts@check@class@options{%
23 \ifpgfopts@process@class
24 \ifx\@classoptionslist\relax\else
25 \expandafter\expandafter\expandafter
26 \pgfopts@check@class@options@aux
27 \fi
28 \fi
29 }
30 \newcommand*\pgfopts@check@class@options@aux{%
31 \@for\pgfopts@current@option:=\@classoptionslist\do
32 {%
33 \pgfkeysifdefined
34 {%
35 \pgfopts@key@path
36 \pgfopts@get@key@name\pgfopts@current@option
37 /.@cmd%
38 }%
39 {%
40 \pgfopts@list@add\pgfopts@options@clist\pgfopts@current@option
41 \@expandtwoargs\@removeelement\pgfopts@current@option
42 \@unusedoptionlist\@unusedoptionlist
43 }%
44 }%
45 }%
46 }

```

`\pgfopts@local@options` 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.

```

47 \newcommand*\pgfopts@local@options{}
48 \newcommand*\pgfopts@process@local@options{%
49   \begingroup
50     \@ifundefined{opt@\@currname.\@currentx}{%
51       {\endgroup}}%
52     {%
53       \toks@\expandafter\expandafter\expandafter
54       {\csname opt@\@currname.\@currentx\endcsname}%
55       \expandafter\endgroup
56       \expandafter\def\expandafter\pgfopts@local@options
57       \expandafter{\the\toks@}%
58       \pgfopts@process@local@options@aux@i
59     }%
60 }
61 \newcommand*\pgfopts@process@local@options@aux@i{%
62   \ifx\@currentx\@clsextension
63     \expandafter\pgfopts@process@local@options@class
64   \else
65     \expandafter\pgfopts@process@local@options@package
66   \fi
67   \ifx\pgfopts@options@clist\@empty\else
68     \expandafter\pgfopts@process@local@options@aux@ii
69   \fi
70 }
71 \newcommand*\pgfopts@process@local@options@aux@ii{%
72   \begingroup
73     \toks@\expandafter{\pgfopts@options@clist}%
74     \edef\pgfopts@options@execute
75     {%
76       \noexpand\pgfkeys
77       {%
78         \pgfopts@key@path .cd,%
79         \the\toks@
80       }%
81     }%
82   \expandafter\endgroup
83   \expandafter\def\expandafter\pgfopts@options@execute
84   \expandafter{\pgfopts@options@execute}%
85 }

```

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.

```

86 \newcommand*\pgfopts@process@local@options@class
87   {%
88     \pgfkeysifdefined{\pgfopts@key@path .unknown/.@cmd}%
89     {\pgfopts@list@add\pgfopts@options@clist\@classoptionslist}
90     {\pgfopts@process@local@options@class@aux}%
91   }
92 \newcommand*\pgfopts@process@local@options@class@aux{%
93   \@for\pgfopts@current@option:=\pgfopts@local@options\do{%
94     \pgfkeysifdefined

```

```

95     {%
96       \pgfopts@key@path
97       \pgfopts@get@key@name\pgfopts@current@option
98       / .@cmd%
99     }%
100    {\pgfopts@list@add\pgfopts@options@clist\pgfopts@current@option}%
101    {\pgfopts@list@add\@unusedoptionlist\pgfopts@current@option}%
102  }%
103 }

```

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

```

104 \newcommand*\pgfopts@process@local@options@package{%
105   \pgfopts@list@add\pgfopts@options@clist\pgfopts@local@options
106 }

```

`\pgfopts@get@key@name` A simple function to leave only the key $\langle name \rangle$ in the input stream, whether there is a $\langle value \rangle$ 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.

```

107 \newcommand\pgfopts@get@key@name[1]{%
108   \expandafter\pgfopts@get@key@name@aux#1=\@nil
109 }
110 \def\pgfopts@get@key@name@aux#1=#2\@nil{#1}

```

`\pgfopts@list@add@a@toks` 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.

```

\pgfopts@list@add@b@toks
\pgfopts@list@add@temp
\pgfopts@list@add
111 \newtoks\pgfopts@list@add@a@toks
112 \newtoks\pgfopts@list@add@b@toks
113 \newcommand*\pgfopts@list@add@temp{}
114 \newcommand\pgfopts@list@add[2]{%
115   \pgfopts@list@add@a@toks\expandafter{#2}%
116   \def\pgfopts@list@add@temp{#2}%
117   \pgfopts@list@add@b@toks\expandafter{#1}%
118   \ifx\pgfopts@options@clist\@empty
119     \edef#1{\the\pgfopts@list@add@a@toks}%
120   \else
121     \ifx\pgfopts@list@add@temp\@empty\else
122       \edef#1%
123         {\the\pgfopts@list@add@b@toks,\the\pgfopts@list@add@a@toks}%
124     \fi
125   \fi
126 }

```

`\ProcessPgfOptions` 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.

```

\ProcessPgfPackageOptions
\pgfopts@star@check
127 \newcommand*\ProcessPgfOptions{%
128   \pgfopts@process@classtrue
129   \pgfopts@star@check
130 }
131 \newcommand*\ProcessPgfPackageOptions{%
132   \pgfopts@process@classfalse
133   \pgfopts@star@check

```

```

134 }
135 \newcommand*\pgfopts@star@check{%
136   \ifstar
137     {%
138       \begingroup
139       \edef\@tempa
140         {%
141           \endgroup
142           \noexpand\pgfopts@process@options{\@currname}%
143         }%
144       \@tempa
145     }%
146     {\pgfopts@process@options}%
147 }
148 \onlypreamble\ProcessPgfOptions
149 \onlypreamble\ProcessPgfPackageOptions
150 </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.

Symbols	D	P
\@classoptionslist 24, 31, 89	\def 10, 11, 12, 56, 83, 110, 116	47, 48, 61, 71, 86, 92, 104, 107, 113, 114, 127, 131, 135
\@clsextension .. 13, 62	\do 31, 93	\newif 5
\@currrent . 13, 50, 54, 62	E	\newtoks 111, 112
\@currname .. 50, 54, 142	\edef ... 74, 119, 122, 139	\noexpand 76, 142
\@empty .. 18, 67, 118, 121	\else 13, 24, 64, 67, 120, 121	
\@expandtwoargs 41	\endcsname 54	P
\@for 31, 93	\endgroup . 51, 55, 82, 141	\pgfkeys 76
\@ifstar 136	\expandafter 14, 25, 53, 55, 56, 57, 63, 65, 68, 73, 82, 83, 84, 108, 115, 117	\pgfkeysifdefined 33, 88, 94
\@ifundefined 50	F	\pgfopts@check@class@options 14, 21
\@nil 108, 110	\fi 15, 27, 28, 66, 69, 124, 125	\pgfopts@check@class@options@aux 21
\@onlypreamble . 148, 149	I	\pgfopts@current@option . 21, 93, 97, 100, 101
\@removeelement 41	\ifpgfopts@process@class 5, 23	\pgfopts@get@key@name 36, 97, 107
\@tempa 139, 144	\ifx 13, 24, 62, 67, 118, 121	\pgfopts@get@key@name@aux 107
\@unprocessedoptions 19	L	\pgfopts@key@path 8, 35, 78, 88, 96
\@unusedoptionlist 42, 101	\let 18, 19	\pgfopts@list@add 40, 89, 100, 101, 105, 111
A	N	\pgfopts@list@add@a@toks 111
\AtEndOfPackage 19	\newcommand 6, 7, 8, 9, 21, 22, 30,	\pgfopts@list@add@b@toks 111
B		
\begingroup . 49, 72, 138		
C		
\csname 54		
\CurrentOption 18		

\pgfopts@list@add@temp 16, 47	\ProcessPgfoptions	3, 127
..... 111	\pgfopts@process@local@options@aux@ii	\ProcessPgfoptions	
\pgfopts@local@options 47	\ProcessPgfoptions	3, 127
..... 47	\pgfopts@process@local@options@aux@ii	\ProvidesPackage 2
\pgfopts@options@clist 47		
... 6, 10, 40, 67,	\pgfopts@process@local@options@class		
73, 89, 100, 105, 118 47	R	
\pgfopts@options@execute	\pgfopts@process@local@options@class@aux...	19, 24	
7, 11, 17, 74, 83, 84 47	\RequirePackage 4
\pgfopts@process@classfalse	\pgfopts@process@local@options@package		
..... 132 47	T	
\pgfopts@process@classtrue	\pgfopts@process@options		
..... 128 8, 142, 146	\the 57, 79, 119, 123
\pgfopts@process@local@options	\pgfopts@star@check	127	\toks@ 53, 57, 73, 79