

# The kvsetkeys package

Heiko Oberdiek  
<heiko.oberdiek at gmail.com>

2011/04/07 v1.13

## Abstract

Package kvsetkeys provides `\kvsetkeys`, a variant of package keyval's `\setkeys`. It allows to specify a handler that deals with unknown options. Active commas and equal signs may be used (e.g. see babel's shorthands) and only one level of curly braces is removed from the values.

## Contents

<b>1</b>	<b>Documentation</b>	<b>2</b>
1.1	Motivation	2
1.2	Normalizing key value lists	3
1.3	Parsing key value lists	3
1.4	Processing key value pairs	4
1.5	Default family handler	4
1.6	Put it all together	4
1.7	Comma separated lists	5
<b>2</b>	<b>Example</b>	<b>5</b>
<b>3</b>	<b>Implementation</b>	<b>6</b>
3.1	Identification	6
3.2	Package loading	8
3.3	Check for $\epsilon$ -TeX	8
3.4	Generic help macros	9
3.5	Normalizing	9
3.6	Parsing key value lists	11
3.7	Parsing comma lists	13
3.8	Processing key value pairs	13
3.9	Error handling	14
3.10	Do it all	14
<b>4</b>	<b>Test</b>	<b>15</b>
4.1	Catcode checks for loading	15
4.2	Macro tests	16
4.2.1	Preamble	16
4.2.2	Time	17
4.2.3	Test sets	17
<b>5</b>	<b>Installation</b>	<b>20</b>
5.1	Download	20
5.2	Bundle installation	21
5.3	Package installation	21
5.4	Refresh file name databases	21
5.5	Some details for the interested	21

<b>6</b>	<b>References</b>	<b>22</b>
<b>7</b>	<b>History</b>	<b>22</b>
	[2006/03/06 v1.0]	22
	[2006/10/19 v1.1]	22
	[2007/09/09 v1.2]	22
	[2007/09/29 v1.3]	23
	[2009/07/19 v1.4]	23
	[2009/07/30 v1.5]	23
	[2009/12/12 v1.6]	23
	[2009/12/22 v1.7]	23
	[2010/01/28 v1.8]	23
	[2010/03/01 v1.9]	23
	[2011/01/30 v1.10]	23
	[2011/03/03 v1.11]	23
	[2011/04/05 v1.12]	23
	[2011/04/07 v1.13]	23
<b>8</b>	<b>Index</b>	<b>24</b>

## 1 Documentation

First I want to recommend the very good review article “A guide to key-value methods” by Joseph Wright [1]. It introduces the different key-value packages and compares them.

### 1.1 Motivation

`\kvsetkeys` serves as replacement for `keyval`’s `\setkeys`. It basically uses the same syntax. But the implementation is more robust and predictable:

**Active syntax characters:** Comma ‘,’ and the equals sign ‘=’ are used inside key value lists as syntax characters. Package `keyval` uses the catcode of the characters that is active during package loading, usually this is catcode 12 (other). But it can happen that the catcode setting of the syntax characters changes. Especially active characters are of interest, because some language adaptations uses them. For example, option `turkish` of package `babel` uses the equals sign as active shorthand character. Therefore package `kvsetkeys` deals with both catcode settings 12 (other) and 13 (active).

**Brace removal:** Package `keyval`’s `\setkeys` removes up to two levels of curly braces around the value in some unpredictable way:

```

\setkeys{fam}{key={{value}}} → value
\setkeys{fam}{key={{value}}} → {value}
\setkeys{fam}{key= {{value}}} → {{value}}

```

This package `kvsetkeys` follows a much stronger rule: Exactly one level of braces are removed from an item, if the item is surrounded by curly braces. An item can be a the key value pair, the key or the value.

```

\kvsetkeys{fam}{key={value}} → value
\kvsetkeys{fam}{key={{value}}} → {value}
\kvsetkeys{fam}{key= {{value}}} → {value}

```

**Arbitrary values:** Unmatched conditionals are supported.

Before I describe `\kvsetkeys` in more detail, first I want to explain, how this package deals with key value lists. For the package also provides low level interfaces that can be used by package authors.

## 1.2 Normalizing key value lists

`\kv@normalize {⟨key value list⟩}`

If the user specifies key value lists, he usually prefers nice formatted source code, e.g.:

```
\hypersetup{
  pdftitle   = {...},
  pdfsubject = {...},
  pdfauthor  = {...},
  pdfkeywords = {...},
  ...
}
```

Thus there can be spaces around keys, around = or around the value. Also empty entries are possible by too many commas. Therefore these spaces and empty entries are silently removed by package `keyval` and this package. Whereas the contents of the value can be protected by curly braces, especially if spaces or commas are used inside, a key name must not use spaces or other syntax characters.

`\kv@normalize` takes a key value list and performs the cleanup:

- Spaces are removed.
- Syntax characters (comma and equal sign) that are active are replaced by the same characters with standard catcode. (Example: `babel`'s language option `turkish` uses the equal sign as active shorthand character.)

The result is stored in `\kv@list`, e.g.:

```
\kv@list → ,pdftitle={...},pdfsubject={...},...
```

Curly braces around values (or keys) remain untouched.

**v1.3+**: One comma is added in front of the list and each pair ends with a comma. Thus an empty list consists of one comma, otherwise two commas encloses the list. Empty entries other than the first are removed.

**v1.0 – v1.2**: Empty entries are removed later. In fact it adds a comma at the begin and end to protect the last value and an easier implementation.

## 1.3 Parsing key value lists

`\kv@parse {⟨key value list⟩} {⟨processor⟩}`

It is easier to parse a normalized list, thus `\kv@parse` normalizes the list and calls `\kv@parse@normalized`.

`\kv@parse@normalized {⟨key value list⟩} {⟨processor⟩}`

Now the key value list is split into single key value pairs. For further processing the key and value are given as arguments for the `⟨processor⟩`:

```
⟨processor⟩ {⟨key⟩} {⟨value⟩}
```

Also key and value are stored in macro names:

- `\kv@key` stores the key.
- `\kv@value` stores the value or if the value was not specified it has the meaning `\relax`.

The behaviour in pseudo code:

```
foreach ( $\langle key \rangle$ ,  $\langle value \rangle$ ) in ( $\langle key\ value\ list \rangle$ )
  \kv@key :=  $\langle key \rangle$ 
  \kv@value :=  $\langle value \rangle$ 
   $\langle processor \rangle$  { $\langle key \rangle$ } { $\langle value \rangle$ }
```

`\kv@break`

Since version 2011/03/03 v1.11 `\kv@break` can be called inside the  $\langle processor \rangle$  of `\kv@parse` or `\kv@parse@normalized`, then the processing is stopped and the following entries discarded.

## 1.4 Processing key value pairs

`\kv@processor@default` { $\langle family \rangle$ } { $\langle key \rangle$ } { $\langle value \rangle$ }

There are many possibilities to process key value pairs. `\kv@processor@default` is the processor used in `\kvsetkeys`. It reimplements and extends the behaviour of keyval's `\setkeys`. In case of unknown keys `\setkeys` raise an error. This processor, however, calls a handler instead, if it is provided by the family. Both  $\langle family \rangle$  and  $\langle key \rangle$  may contain package `babel`'s shorthands (since 2011/04/07 v1.13).

The behaviour in pseudo code:

```
if  $\langle key \rangle$  exists
  call the keyval code of  $\langle key \rangle$ 
else
  if  $\langle handler \rangle$  for  $\langle family \rangle$  exists
     $\langle handler \rangle$  { $\langle key \rangle$ } { $\langle value \rangle$ }
  else
    raise unknown key error
fi
fi
```

## 1.5 Default family handler

`\kv@processor@default` calls  $\langle handler \rangle$ , the default handler for the family, if the key does not exist in the family. The handler is called with two arguments, the key and the value. It can be defined with `\kv@set@family@handler`:

`\kv@set@family@handler` { $\langle family \rangle$ } { $\langle handler\ definition \rangle$ }

This sets the default family handler for the keyval family  $\langle family \rangle$ . Inside  $\langle handler\ definition \rangle$  #1 stands for the key and #2 is the value. Also `\kv@key` and `\kv@value` can be used for the key and the value. If the value is not given, `\kv@value` has the meaning `\relax`.

## 1.6 Put it all together

`\kvsetkeys` { $\langle family \rangle$ } { $\langle key\ value\ list \rangle$ }

The work is done by the previous commands. `\kvsetkeys` just calls them:

```
\kv@parse { $\langle key\ value\ list \rangle$ } {\kv@processor@default { $\langle family \rangle$ }}
```

Thus you can replace `\setkeys` of package `keyval` by the key value parser of this package:

```
\renewcommand*\setkeys{\kvsetkeys}
or
\let\setkeys\kvsetkeys
```

## 1.7 Comma separated lists

Since version 2007/09/29 v1.3 this package also supports the normalizing and parsing of general comma separated lists.

`\comma@normalize {<comma list>}`

Macro `\comma@normalize` normalizes the comma separated list, removes spaces around commas. The result is put in macro `\comma@list`.

`\comma@parse {<comma list>} {<processor>}`

Macro `\comma@parse` first normalizes the comma separated list and then parses the list by calling `\comma@parse@normalized`.

`\comma@parse@normalized {<normalized comma list>} {<processor>}`

The list is parsed. Empty entries are ignored. `<processor>` is called for each non-empty entry with the entry as argument:

```
<processor>{<entry>}
```

Also the entry is stored in the macro `\comma@entry`.

`\comma@break`

Since version 2011/03/03 v1.11 `\comma@break` can be called inside the `<processor>` of `\comma@parse` or `\comma@parse@normalized`, then the processing is stopped and the following entries discarded.

## 2 Example

The following example prints a short piece of HTML code using the tabbing environment for indenting purpose and a key value syntax for specifying the attributes of an HTML tag. The example illustrates the use of a default family handler.

```
1 <*example>
2 \documentclass{article}
3 \usepackage[T1]{fontenc}
4 \usepackage{kvsetkeys}
5 \usepackage{keyval}
6
7 \makeatletter
8 \newcommand*{\tag}[2] [] {%
9 % #1: attributes
10 % #2: tag name
11 \begingroup
12 \toks@={}%
13 \let\@endslash\@empty
14 \kvsetkeys{tag}{#1}%
15 \texttt{%
16 \textless #2\the\toks@\@endslash\textgreater
```

```

17   }%
18   \endgroup
19 }
20 \kv@set@family@handler{tag}{%
21   % #1: key
22   % #2: value
23   \toks@\expandafter{%
24     \the\toks@
25     \space
26     #1=\string"#2\string"%
27   }%
28 }
29 \define@key{tag}{/}[]{}%
30 \def\@endslash{/}%
31 }
32 \makeatother
33
34 \begin{document}
35 \begin{tabbing}
36   \mbox{} \quad = \quad \kill
37   \tag{html} \backslash
38   \> \dots \backslash
39   \> \tag[border=1]{table} \backslash
40   \> \> \tag[width=200, span=3, /]{colgroup} \backslash
41   \> \> \dots \backslash
42   \> \tag{/table} \backslash
43   \> \dots \backslash
44   \tag{/html} \backslash
45 \end{tabbing}
46 \end{document}
47 \end{example}

```

## 3 Implementation

### 3.1 Identification

```
48 \*package)
```

Reload check, especially if the package is not used with L<sup>A</sup>T<sub>E</sub>X.

```

49 \begingroup\catcode61\catcode48\catcode32=10\relax%
50   \catcode13=5 % ^~M
51   \endlinechar=13 %
52   \catcode35=6 % #
53   \catcode39=12 % '
54   \catcode44=12 % ,
55   \catcode45=12 % -
56   \catcode46=12 % .
57   \catcode58=12 % :
58   \catcode64=11 % @
59   \catcode123=1 % {
60   \catcode125=2 % }
61   \expandafter\let\expandafter\x\csname ver@kvsetkeys.sty\endcsname
62   \ifx\x\relax % plain-TeX, first loading
63   \else
64     \def\empty{}%
65     \ifx\x\empty % LaTeX, first loading,
66       % variable is initialized, but \ProvidesPackage not yet seen
67     \else
68       \expandafter\ifx\csname PackageInfo\endcsname\relax
69         \def\x#1#2{%
70           \immediate\write-1{Package #1 Info: #2.}%
71         }%

```

```

72     \else
73     \def\x#1#2{\PackageInfo{#1}{#2, stopped}}%
74     \fi
75     \x{kvsetkeys}{The package is already loaded}%
76     \aftergroup\endinput
77     \fi
78 \fi
79 \endgroup%

Package identification:
80 \begingroup\catcode61\catcode48\catcode32=10\relax%
81 \catcode13=5 % ^~M
82 \endlinechar=13 %
83 \catcode35=6 % #
84 \catcode39=12 % '
85 \catcode40=12 % (
86 \catcode41=12 % )
87 \catcode44=12 % ,
88 \catcode45=12 % -
89 \catcode46=12 % .
90 \catcode47=12 % /
91 \catcode58=12 % :
92 \catcode64=11 % @
93 \catcode91=12 % [
94 \catcode93=12 % ]
95 \catcode123=1 % {
96 \catcode125=2 % }
97 \expandafter\ifx\csname ProvidesPackage\endcsname\relax
98   \def\x#1#2#3[#4]{\endgroup
99     \immediate\write-1{Package: #3 #4}%
100    \xdef#1{#4}%
101   }%
102 \else
103   \def\x#1#2[#3]{\endgroup
104     #2[#{#3}]%
105     \ifx#1@undefined
106       \xdef#1{#3}%
107     \fi
108     \ifx#1\relax
109       \xdef#1{#3}%
110     \fi
111   }%
112 \fi
113 \expandafter\x\csname ver@kvsetkeys.sty\endcsname
114 \ProvidesPackage{kvsetkeys}%
115 [2011/04/07 v1.13 Key value parser (HO)]%
116 \begingroup\catcode61\catcode48\catcode32=10\relax%
117 \catcode13=5 % ^~M
118 \endlinechar=13 %
119 \catcode123=1 % {
120 \catcode125=2 % }
121 \catcode64=11 % @
122 \def\x{\endgroup
123   \expandafter\edef\csname KVS@AtEnd\endcsname{%
124     \endlinechar=\the\endlinechar\relax
125     \catcode13=\the\catcode13\relax
126     \catcode32=\the\catcode32\relax
127     \catcode35=\the\catcode35\relax
128     \catcode61=\the\catcode61\relax
129     \catcode64=\the\catcode64\relax
130     \catcode123=\the\catcode123\relax
131     \catcode125=\the\catcode125\relax
132   }%

```

```

133 }%
134 \x\catcode61\catcode48\catcode32=10\relax%
135 \catcode13=5 % ^^M
136 \endlinechar=13 %
137 \catcode35=6 % #
138 \catcode64=11 % @
139 \catcode123=1 % {
140 \catcode125=2 % }
141 \def\TMP@EnsureCode#1#2{%
142   \edef\KVS@AtEnd{%
143     \KVS@AtEnd
144     \catcode#1=\the\catcode#1\relax
145   }%
146   \catcode#1=#2\relax
147 }
148 \TMP@EnsureCode{36}{3}% $
149 \TMP@EnsureCode{38}{4}% &
150 \TMP@EnsureCode{39}{12}% '
151 \TMP@EnsureCode{44}{12}% ,
152 \TMP@EnsureCode{46}{12}% .
153 \TMP@EnsureCode{47}{12}% /
154 \TMP@EnsureCode{91}{12}% [
155 \TMP@EnsureCode{93}{12}% ]
156 \TMP@EnsureCode{94}{7}% ^ (superscript)
157 \TMP@EnsureCode{96}{12}% '
158 \TMP@EnsureCode{126}{13}% ~ (active)
159 \edef\KVS@AtEnd{\KVS@AtEnd\noexpand\endinput}

```

### 3.2 Package loading

```

160 \begingroup\expandafter\expandafter\expandafter\endgroup
161 \expandafter\ifx\csname RequirePackage\endcsname\relax
162   \def\TMP@RequirePackage#1[#2]{%
163     \begingroup\expandafter\expandafter\expandafter\endgroup
164     \expandafter\ifx\csname ver@#1.sty\endcsname\relax
165       \input #1.sty\relax
166     \fi
167   }%
168   \TMP@RequirePackage{infwarerr}[2007/09/09]%
169   \TMP@RequirePackage{etexcmds}[2010/01/28]%
170 \else
171   \RequirePackage{infwarerr}[2007/09/09]%
172   \RequirePackage{etexcmds}[2010/01/28]%
173 \fi

174 \expandafter\ifx\csname toks@\endcsname\relax
175   \toksdef\toks@=0 %
176 \fi

```

### 3.3 Check for $\epsilon$ -TeX

\unexpanded, \ifcsname, and \unless are used if found.

```

177 \begingroup\expandafter\endgroup
178 \ifcase0\ifetex@unexpanded
179   \expandafter\ifx\csname ifcsname\endcsname\relax
180   \else
181     \expandafter\ifx\csname unless\endcsname\relax
182     \else
183       1%
184     \fi
185   \fi
186 \fi
187 \catcode'\$=9 % ignore
188 \catcode'\&=14 % comment

```



```

189 \else % e-TeX
190 \catcode'\$=14 % comment
191 \catcode'\&=9 % ignore
192 \fi

```

### 3.4 Generic help macros

```

\KVS@Empty
193 \def\KVS@Empty{}

\KVS@FirstOfTwo
194 \long\def\KVS@FirstOfTwo#1#2{#1}

\KVS@SecondOfTwo
195 \long\def\KVS@SecondOfTwo#1#2{#2}

\KVS@IfEmpty
196 \long\def\KVS@IfEmpty#1{%
197 & \edef\KVS@Temp{\etex@unexpanded{#1}}%
198 $ \begingroup
199 $ \toks@{#1}%
200 $ \edef\KVS@Temp{\the\toks@}%
201 $ \expandafter\endgroup
202 \ifx\KVS@Temp\KVS@Empty
203 \expandafter\KVS@FirstOfTwo
204 \else
205 \expandafter\KVS@SecondOfTwo
206 \fi
207 }

```

### 3.5 Normalizing

```

\kv@normalize
208 \long\def\kv@normalize#1{%
209 \begingroup
210 \toks@{,#1,}%
211 \KVS@Comma
212 \KVS@SpaceComma{ }%
213 \KVS@CommaSpace
214 \KVS@CommaComma
215 \KVS@Equals
216 \KVS@SpaceEquals{ }%
217 \KVS@EqualsSpace
218 \xdef\KVS@Global{\the\toks@}%
219 \endgroup
220 \let\kv@list\KVS@Global
221 }

\comma@normalize
222 \def\comma@normalize#1{%
223 \begingroup
224 \toks@{,#1,}%
225 \KVS@Comma
226 \KVS@SpaceComma{ }%
227 \KVS@CommaSpace
228 \KVS@CommaComma
229 \xdef\KVS@Global{\the\toks@}%
230 \endgroup
231 \let\comma@list\KVS@Global
232 }

```

`\KVS@Comma` Converts active commas into comma with catcode other. Also adds a comma at the end to protect the last value for next cleanup steps.

```

233 \begingroup
234 \lccode'\,='\",%
235 \lccode'\~='\",%
236 \lowercase{\endgroup
237 \def\KVS@Comma{%
238 \toks@\expandafter{\expandafter}\expandafter
239 \KVS@@Comma\the\toks@\KVS@Nil
240 }%
241 \long\def\KVS@@Comma#1~#2\KVS@Nil{%
242 \toks@\expandafter{\the\toks@#1}%
243 \KVS@IfEmpty{#2}{%
244 }{%
245 \KVS@@Comma,#2\KVS@Nil
246 }%
247 }%
248 }

```

`\KVS@SpaceComma` Removes spaces before the comma, may add commas at the end.

```

249 \long\def\KVS@SpaceComma#1{%
250 \expandafter\KVS@@SpaceComma\the\toks@#1,\KVS@Nil
251 }

```

`\KVS@@SpaceComma`

```

252 \long\def\KVS@@SpaceComma#1 ,#2\KVS@Nil{%
253 \KVS@IfEmpty{#2}{%
254 \toks@{#1}%
255 }{%
256 \toks@{#1,#2}%
257 \expandafter\KVS@@SpaceComma\the\toks@\KVS@Nil
258 }%
259 }

```

`\KVS@CommaSpace` Removes spaces after the comma, may add commas at the end.

```

260 \def\KVS@CommaSpace{%
261 \expandafter\KVS@@CommaSpace\the\toks@, \KVS@Nil
262 }

```

`\KVS@@CommaSpace`

```

263 \long\def\KVS@@CommaSpace#1 ,#2\KVS@Nil{%
264 \KVS@IfEmpty{#2}{%
265 \toks@{#1}%
266 }{%
267 \toks@{#1,#2}%
268 \expandafter\KVS@@CommaSpace\the\toks@\KVS@Nil
269 }%
270 }

```

`\KVS@CommaComma` Replaces multiple commas by one comma.

```

271 \def\KVS@CommaComma{%
272 \expandafter\KVS@@CommaComma\the\toks@,\KVS@Nil
273 }

```

`\KVS@@CommaComma`

```

274 \long\def\KVS@@CommaComma#1 , ,#2\KVS@Nil{%
275 \toks@{#1,#2}%
276 \KVS@IfEmpty{#2}{%
277 }{%
278 \expandafter\KVS@@CommaComma\the\toks@\KVS@Nil
279 }%
280 }

```

`\KVS@Equals` Converts active equals signs into catcode other characters.

```
281 \begingroup
282 \lccode'\=='\=%
283 \lccode'\~='\=%
284 \lowercase{\endgroup
285 \def\KVS@Equals{%
286 \toks@\expandafter{\expandafter}\expandafter
287 \KVS@@Equals\the\toks@\KVS@Nil
288 }%
289 \long\def\KVS@@Equals#1~#2\KVS@Nil{%
290 \edef\KVS@Temp{\the\toks@}%
291 \ifx\KVS@Temp\KVS@Empty
292 \expandafter\KVS@FirstOfTwo
293 \else
294 \expandafter\KVS@SecondOfTwo
295 \fi
296 {%
297 \toks@{#1}%
298 }{%
299 \toks@\expandafter{\the\toks@=#1}%
300 }%
301 \KVS@IfEmpty{#2}{%
302 }{%
303 \KVS@@Equals#2\KVS@Nil
304 }%
305 }%
306 }
```

`\KVS@SpaceEquals` Removes spaces before the equals sign.

```
307 \long\def\KVS@SpaceEquals#1{%
308 \expandafter\KVS@@SpaceEquals\the\toks@#1=\KVS@Nil
309 }
```

`\KVS@@SpaceEquals`

```
310 \long\def\KVS@@SpaceEquals#1=#2\KVS@Nil{%
311 \KVS@IfEmpty{#2}{%
312 \toks@{#1}%
313 }{%
314 \toks@{#1=#2}%
315 \expandafter\KVS@@SpaceEquals\the\toks@\KVS@Nil
316 }%
317 }
```

`\KVS@EqualsSpace` Removes spaces after the equals sign.

```
318 \def\KVS@EqualsSpace{%
319 \expandafter\KVS@@EqualsSpace\the\toks@=\KVS@Nil
320 }
```

`\KVS@@EqualsSpace`

```
321 \long\def\KVS@@EqualsSpace#1=#2\KVS@Nil{%
322 \KVS@IfEmpty{#2}{%
323 \toks@{#1}%
324 }{%
325 \toks@{#1=#2}%
326 \expandafter\KVS@@EqualsSpace\the\toks@\KVS@Nil
327 }%
328 }
```

### 3.6 Parsing key value lists

`\kv@parse` Normalizes and parses the key value list. Also sets `\kv@list`.

```

329 \long\def\kv@parse#1{%
330   \kv@normalize{#1}%
331   \expandafter\kv@parse@normalized\expandafter{\kv@list}%
332 }

\kv@parse@normalized #1: key value list
#2: processor
333 \long\def\kv@parse@normalized#1#2{%
334   \KVS@Parse#1,\KVS@Nil{#2}%
335 }

\KVS@Parse #1,#2: key value list
#3: processor
336 \long\def\KVS@Parse#1,#2\KVS@Nil#3{%
337   \KVS@IfEmpty{#1}{%
338     }{%
339     \KVS@Process#1=\KVS@Nil{#3}%
340   }%
341   \KVS@MaybeBreak
342   \KVS@IfEmpty{#2}{%
343     }{%
344     \KVS@Parse#2\KVS@Nil{#3}%
345   }%
346 }

\KVS@Process #1: key
#2: value, =
#3: processor
347 \long\def\KVS@Process#1=#2\KVS@Nil#3{%
348   \let\KVS@MaybeBreak\relax
349   \def\kv@key{#1}%
350   \KVS@IfEmpty{#2}{%
351     \let\kv@value\relax
352     #3{#1}{}%
353   }{%
354     \KVS@@Process{#1}#2\KVS@Nil{#3}%
355   }%
356 }

\KVS@@Process #1: key
#2: value
#3: processor
357 \long\def\KVS@@Process#1#2=\KVS@Nil#3{%
358   & \edef\kv@value{\etex@unexpanded{#2}}%
359   $ \begingroup
360   $ \toks@{#2}%
361   $ \xdef\KVS@Global{\the\toks@}%
362   $ \endgroup
363   $ \let\kv@value\KVS@Global
364   #3{#1}{#2}%
365 }

\KVS@MaybeBreak
366 \let\KVS@MaybeBreak\relax

\KVS@break
367 \def\KVS@break#1#2#3#4{%
368   \let\KVS@MaybeBreak\relax
369 }

```

`\kv@break`

```
370 \def\kv@break{%
371   \let\KVS@MaybeBreak\KVS@break
372 }
```

### 3.7 Parsing comma lists

`\comma@parse` Normalizes and parses the key value list. Also sets `\comma@list`.

```
373 \def\comma@parse#1{%
374   \comma@normalize{#1}%
375   \expandafter\comma@parse@normalized\expandafter{\comma@list}%
376 }
```

`\comma@parse@normalized` #1: comma list  
#2: processor

```
377 \def\comma@parse@normalized#1#2{%
378   \KVS@CommaParse#1,\KVS@Nil{#2}%
379 }
```

`\KVS@CommaParse` #1,#2: comma list  
#3: processor

```
380 \def\KVS@CommaParse#1,#2\KVS@Nil#3{%
381   \KVS@IfEmpty{#1}{%
382     }{%
383       \def\comma@entry{#1}%
384       #3{#1}%
385     }%
386   \KVS@MaybeBreak
387   \KVS@IfEmpty{#2}{%
388     }{%
389     \KVS@CommaParse#2\KVS@Nil{#3}%
390   }%
391 }
```

`\comma@break`

```
392 \def\comma@break{%
393   \let\KVS@MaybeBreak\KVS@break
394 }
```

### 3.8 Processing key value pairs

`\kv@processor@default`

```
395 \def\kv@processor@default#1#2#3{%
396   \begingroup
397     \csname @safe@activetrue\endcsname
398     \let\ifincsname\iftrue
399     \edef\KVS@temp{\endgroup
400       \noexpand\KVS@ProcessorDefault{#1}{#2}%
401     }%
402   \KVS@temp
403 }
```

`\KVS@ProcessorDefault`

```
404 \long\def\KVS@ProcessorDefault#1#2#3{%
405   & \unless\ifincsname KV@#1@#2\endcsname
406   $ \begingroup\expandafter\expandafter\expandafter\endgroup
407   $ \expandafter\ifx\csname KV@#1@#2\endcsname\relax
408   & \unless\ifincsname KVS@#1@handler\endcsname
409   $ \begingroup\expandafter\expandafter\expandafter\endgroup
410   $ \expandafter\ifx\csname KVS@#1@handler\endcsname\relax
```

```

411     \kv@error@unknownkey{#1}{#2}%
412   \else
413     \csname KVS@#1@handler\endcsname{#2}{#3}%
414     \relax
415   \fi
416 \else
417   \ifx\kv@value\relax
418 &   \unless\ifcsname KV@#1@#2@default\endcsname
419 $   \begingroup\expandafter\expandafter\expandafter\endgroup
420 $   \expandafter\ifx\csname KV@#1@#2@default\endcsname\relax
421     \kv@error@novalue{#1}{#2}%
422   \else
423     \csname KV@#1@#2@default\endcsname
424     \relax
425   \fi
426 \else
427   \csname KV@#1@#2\endcsname{#3}%
428 \fi
429 \fi
430 }

```

`\kv@set@family@handler`

```

431 \long\def\kv@set@family@handler#1#2{%
432   \begingroup
433   \csname @safe@activetrue\endcsname
434   \let\ifincsname\iftrue
435   \expandafter\endgroup
436   \expandafter\def\csname KVS@#1@handler\endcsname##1##2{#2}%
437 }

```

### 3.9 Error handling

`\kv@error@novalue`

```

438 \def\kv@error@novalue{%
439   \kv@error@generic{No value specified for}%
440 }

```

`\kv@error@unknownkey`

```

441 \def\kv@error@unknownkey{%
442   \kv@error@generic{Undefined}%
443 }

```

`\kv@error@generic`

```

444 \def\kv@error@generic#1#2#3{%
445   \@PackageError{kvsetkeys}{%
446     #1 key ‘#3’%
447   }{%
448     The keyval family of the key ‘#3’ is ‘#2’.\MessageBreak
449     The setting of the key is ignored because of the error.\MessageBreak
450     \MessageBreak
451     \@ehc
452   }%
453 }

```

### 3.10 Do it all

`\kvsetkeys`

```

454 \long\def\kvsetkeys#1#2{%
455   \kv@parse{#2}{\kv@processor@default{#1}}%
456 }

```

```

457 \KVS@AtEnd%
458 </package>

```

## 4 Test

### 4.1 Catcode checks for loading

```

459 (*test1)
460 \catcode'\{=1 %
461 \catcode'\}=2 %
462 \catcode'\#=6 %
463 \catcode'\@=11 %
464 \expandafter\ifx\csname count@\endcsname\relax
465   \countdef\count@=255 %
466 \fi
467 \expandafter\ifx\csname @gobble\endcsname\relax
468   \long\def@gobble#1{%
469 \fi
470 \expandafter\ifx\csname @firstofone\endcsname\relax
471   \long\def@firstofone#1{#1}%
472 \fi
473 \expandafter\ifx\csname loop\endcsname\relax
474   \expandafter@firstofone
475 \else
476   \expandafter@gobble
477 \fi
478 {%
479   \def\loop#1\repeat{%
480     \def\body{#1}%
481     \iterate
482   }%
483   \def\iterate{%
484     \body
485     \let\next\iterate
486   \else
487     \let\next\relax
488   \fi
489   \next
490 }%
491 \let\repeat=\fi
492 }%
493 \def\RestoreCatcodes{}
494 \count@=0 %
495 \loop
496   \edef\RestoreCatcodes{%
497     \RestoreCatcodes
498     \catcode\the\count@=\the\catcode\count@\relax
499   }%
500 \ifnum\count@<255 %
501   \advance\count@ 1 %
502 \repeat
503
504 \def\RangeCatcodeInvalid#1#2{%
505   \count@=#1\relax
506   \loop
507     \catcode\count@=15 %
508   \ifnum\count@<#2\relax
509     \advance\count@ 1 %
510   \repeat
511 }
512 \def\RangeCatcodeCheck#1#2#3{%

```

```

513 \count@=#1\relax
514 \loop
515   \ifnum#3=\catcode\count@
516   \else
517     \errmessage{%
518       Character \the\count@\space
519       with wrong catcode \the\catcode\count@\space
520       instead of \number#3%
521     }%
522   \fi
523 \ifnum\count@<#2\relax
524   \advance\count@ 1 %
525 \repeat
526 }
527 \def\space{ }
528 \expandafter\ifx\csname LoadCommand\endcsname\relax
529   \def\LoadCommand{\input kvsetkeys.sty\relax}%
530 \fi
531 \def\Test{%
532   \RangeCatcodeInvalid{0}{47}%
533   \RangeCatcodeInvalid{58}{64}%
534   \RangeCatcodeInvalid{91}{96}%
535   \RangeCatcodeInvalid{123}{255}%
536   \catcode'\@=12 %
537   \catcode'\=0 %
538   \catcode'\%=14 %
539   \LoadCommand
540   \RangeCatcodeCheck{0}{36}{15}%
541   \RangeCatcodeCheck{37}{37}{14}%
542   \RangeCatcodeCheck{38}{47}{15}%
543   \RangeCatcodeCheck{48}{57}{12}%
544   \RangeCatcodeCheck{58}{63}{15}%
545   \RangeCatcodeCheck{64}{64}{12}%
546   \RangeCatcodeCheck{65}{90}{11}%
547   \RangeCatcodeCheck{91}{91}{15}%
548   \RangeCatcodeCheck{92}{92}{0}%
549   \RangeCatcodeCheck{93}{96}{15}%
550   \RangeCatcodeCheck{97}{122}{11}%
551   \RangeCatcodeCheck{123}{255}{15}%
552   \RestoreCatcodes
553 }
554 \Test
555 \csname @@end\endcsname
556 \end
557 </test1>

```

## 4.2 Macro tests

### 4.2.1 Preamble

```

558 <test2>
559 \NeedsTeXFormat{LaTeX2e}
560 \nofiles
561 \documentclass{article}
562 (noetex) \let\SavedUnexpanded\unexpanded
563 (noetex) \let\unexpanded\UNDEFINED
564 \makeatletter
565 \chardef\KVS@TestMode=1 %
566 \makeatother
567 \usepackage{kvsetkeys}[2011/04/07]
568 (noetex) \let\unexpanded\SavedUnexpanded
569 \usepackage{qstest}
570 \IncludeTests{*}

```



```
571 \LogTests{log}{*}{*}
```

#### 4.2.2 Time

```
572 \begingroup\expandafter\expandafter\expandafter\endgroup
573 \expandafter\ifx\csname pdfresettimer\endcsname\relax
574 \else
575 \makeatletter
576 \newcount\SummaryTime
577 \newcount\TestTime
578 \SummaryTime=\z@
579 \newcommand*\PrintTime[2]{%
580   \typeout{%
581     [Time #1: \strip@pt\dimexpr\number#2sp\relax\space s]%
582   }%
583 }%
584 \newcommand*\StartTime[1]{%
585   \renewcommand*\TimeDescription{#1}%
586   \pdfresettimer
587 }%
588 \newcommand*\TimeDescription{}%
589 \newcommand*\StopTime{%
590   \TestTime=\pdfelapsedtime
591   \global\advance\SummaryTime\TestTime
592   \PrintTime\TimeDescription\TestTime
593 }%
594 \let\saved@qstest\qstest
595 \let\saved@endqstest\endqstest
596 \def\qstest#1#2{%
597   \saved@qstest{#1}{#2}%
598   \StartTime{#1}%
599 }%
600 \def\endqstest{%
601   \StopTime
602   \saved@endqstest
603 }%
604 \AtEndDocument{%
605   \PrintTime{summary}\SummaryTime
606 }%
607 \makeatother
608 \fi
```

#### 4.2.3 Test sets

```
609 \makeatletter
610 \def\@makeactive#1{%
611   \catcode'#1=13\relax
612 }
613 \@makeactive\,
614 \def,{\errmessage{COMMA}}
615 \@makeother\,
616 \@makeactive\=
617 \def={\errmessage{EQUALS}}
618 \@makeother\=
619
620 \begin{qstest}{normalize}{normalize,active-chars,space-removal}%
621   \long\def\Test#1#2{%
622     \@makeother\,%
623     \@makeother\=%
624     \scantokens{\toks@={#2}}%
625     \edef\Result{\the\toks@}%
626     \@makeother\,%
627     \@makeother\=%
628     \@Test{#1}%
629     \@makeactive\,%
```

```

630 \@Test{#1}%
631 \@makeactive\=%
632 \@Test{#1}%
633 \@makeother\,%
634 \@Test{#1}%
635 \@makeother\=%
636 }%
637 \long\def\@Test#1{%
638 \scantokens{\kv@normalize{#1}}%
639 \expandafter\expandafter\expandafter\Expect
640 \expandafter\expandafter\expandafter
641 {\expandafter\kv@list\expandafter}\expandafter{\Result}%
642 \Expect*{\ifx\kv@list\Result true\else false\fi}{true}%
643 }%
644 \Test{}{,}%
645 \Test{,}{,}%
646 \Test{,,}{,}%
647 \Test{,,}{,}%
648 \Test{ , }{,}%
649 \Test{{a}}{, {a},}%
650 \Test{, {a}}{, {a},}%
651 \Test{{a},}{, {a},}%
652 \Test{{a}, {b}}{, {a}, {b},}%
653 \Test{{b}={c}, {}=, {}={}, {d}={}, {b}={c}, {}=, {}={}, {d}={},}%
654 \Test{{}}{, {}},%
655 \Test{{}, {}, {}}{, {}, {}, {},}%
656 \Test{=} {, =,}%
657 \Test{=, =, =} {, =, =, =,}%
658 \Test{a=\par} {, a=\par,}%
659 \Test{\par} {, \par,}%
660 \def\TestSet#1{%
661 \Test{#1#1}{,}%
662 \Test{#1#1, #1#1}{,}%
663 \Test{#1#1, #1#1, #1#1}{,}%
664 \Test{#1#1#1#1#1}{,}%
665 \Test{{a}#1#1=#1#1{b}} {, {a}={b},}%
666 }%
667 \TestSet{ }%
668 \begingroup
669 \let\saved@normalize\kv@normalize
670 \def\kv@normalize#1{%
671 \saved@normalize{#1}%
672 \@onelevel@sanitize\kv@list
673 \@onelevel@sanitize\Result
674 }%
675 \Test{#, ##, {#}={#}, {#}=#, {#}} {, #, ##, {#}={#}, {#}=#, {#},}%
676 \endgroup
677 \begingroup
678 \def\Test#1#2{%
679 \edef\Result{#2}%
680 \@Test{#1}%
681 }%
682 \Test{{ a = b }} {, { a = b },}%
683 \@makeactive\,%
684 \Test{{,}} {\string, {\noexpand,} \string,}%
685 \@makeother\,%
686 \@makeactive\=%
687 \Test{a={}} {, a\string={\noexpand=},}%
688 \endgroup
689 \Test{a=b} {, a=b,}%
690 \Test{a={b}} {, a={b},}%
691 \Test{a = {b}} {, a={b},}%

```

```

692 \Test{a= {b}}{,a={b},}%
693 \Test{a = {b}}{,a={b},}%
694 \Test{a = {b} ,}{,a={b},}%
695 \Test{a}{,a,}%
696 \Test{ a}{,a,}%
697 \Test{a }{,a,}%
698 \Test{ a }{,a,}%
699 \Test{, a ,}{,a,}%
700 \Test{, a b ,}{,a b,}%
701 \Test{,a ,}{,a,}%
702 \Test{ a =}{,a=,}%
703 \Test{ a = }{,a=,}%
704 \Test{a =}{,a=,}%
705 \Test{{a} =}{,{a}=,}%
706 \Test{{a}= {}}{,{a}={},}%
707 \Test{, a = {}}{,a={},}%
708 \Test{a, ,b}{,a,b,}%
709 \Test{a=\fi}{,a=\fi,}%
710 \Test{a=\iffalse}{,a=\iffalse,}%
711 \Test{a=\iffalse,b=\fi}{,a=\iffalse,b=\fi,}%
712 \end{qstest}
713
714 \begin{qstest}{parse}{parse,brace-removal}
715 \def\Processor#1#2{%
716 \expandafter\Expect\expandafter{\kv@key}{#1}%
717 \toks@{#2}%
718 \edef\x{\the\toks@}%
719 \ifx\kv@value\relax
720 \Expect*{\the\toks@}{}%
721 \def\Value{<>}%
722 \else
723 \edef\Value{[\the\toks@]}%
724 \@onelevel@sanitize\Value
725 \fi
726 \toks@{#1}%
727 \ifx\Result\@empty
728 \edef\Result{[\the\toks@]=\Value}%
729 \else
730 \edef\Result{\Result, [\the\toks@]=\Value}%
731 \fi
732 \@onelevel@sanitize\Result
733 }%
734 \def\Test#1#2{%
735 \sbox0{%
736 \let\Result\@empty
737 \kv@parse{#1}\Processor
738 \Expect*{\Result}{#2}%
739 }%
740 \Expect*{\the\wd0}{0.0pt}%
741 }%
742 \Test{}{}%
743 \Test{{}}{}%
744 \Test{{{}}}{[ ]=<>}%
745 \Test{{{}}}{[{}]=<>}%
746 \Test{a}{[a]=<>}%
747 \Test{{a}}{[a]=<>}%
748 \Test{{{a}}}{[a]=<>}%
749 \Test{{{a}}}{[a]=<>}%
750 \Test{{{a}}}{[{}a]=<>}%
751 \Test{a=}{[a]=[ ]}%
752 \Test{{a}=}{[a]=[ ]}%
753 \Test{{{a}=}{[{}a]=[ ]}%

```

```

754 \Test{a={}}{[a=[]}%
755 \Test{{a}={}}{[a]=[{}]}%
756 \Test{a=b}{[a]=[b]}%
757 \Test{a=\fi}{[a]=[\fi]}%
758 \Test{a=\iffalse}{[a]=[\iffalse]}%
759 \Test{a=\iffalse,b=\fi}{[a]=[\iffalse],[b]=[\fi]}%
760 \Test{{ a = b }}{[ a ]=[ b ]}%
761 \Test{{{ a = b }}}{[ a = b ]=<>}%
762 \end{qstest}
763
764 \begin{qstest}{comma}{comma,parse}
765 \def\Processor#1{%
766   \expandafter\Expect\expandafter{\comma@entry}{#1}%
767   \toks@{#1}%
768   \ifx\Result\@empty
769     \edef\Result{[\the\toks@]}%
770   \else
771     \edef\Result{\Result,[\the\toks@]}%
772   \fi
773   \@onelevel@sanitize\Result
774 }%
775 \def\Test#1#2{%
776   \sbox0{%
777     \let\Result\@empty
778     \comma@parse{#1}\Processor
779     \Expect*{\Result}{#2}%
780   }%
781   \Expect*{\the\wd0}{0.0pt}%
782 }%
783 \Test{}{}%
784 \Test{{}}{}%
785 \Test{{{}}{[{}]}%
786 \Test{a}{[a]}%
787 \Test{{a}}{[a]}%
788 \Test{{{a}}{[[a]]}%
789 \Test{a=}{[a=]}%
790 \Test{a\fi}{[a\fi]}%
791 \Test{a\iffalse}{[a\iffalse]}%
792 \Test{\iffalse,\fi}{[\iffalse],[\fi]}%
793 \Test{ a , b , c }{[a],[b],[c]}%
794 \Test{ { } , { } , { } , { } , { } }{[ ],[ ],[ ],[ ],[ ]}%
795 \Test{ {{ } , {{ } , {{ } , {{ } , {{ } } } } }{[{}],[{}],[{}],[{}],[{}]}%
796 \end{qstest}
797
798 \begin{document}
799 \end{document}
800 </test2>

```

## 5 Installation

### 5.1 Download

**Package.** This package is available on CTAN<sup>1</sup>:

[CTAN:macros/latex/contrib/oberdiek/kvsetkeys.dtx](https://ctan.org/ctan/packages/macros/latex/contrib/oberdiek/kvsetkeys.dtx) The source file.

[CTAN:macros/latex/contrib/oberdiek/kvsetkeys.pdf](https://ctan.org/ctan/packages/macros/latex/contrib/oberdiek/kvsetkeys.pdf) Documentation.

**Bundle.** All the packages of the bundle ‘oberdiek’ are also available in a TDS compliant ZIP archive. There the packages are already unpacked and the documentation files are generated. The files and directories obey the TDS standard.

<sup>1</sup><http://ftp.ctan.org/tex-archive/>

[CTAN:install/macros/latex/contrib/oberdiek.tds.zip](#)

*TDS* refers to the standard “A Directory Structure for  $\TeX$  Files” ([CTAN:tds/tds.pdf](#)). Directories with `texmf` in their name are usually organized this way.

## 5.2 Bundle installation

**Unpacking.** Unpack the `oberdiek.tds.zip` in the TDS tree (also known as `texmf` tree) of your choice. Example (linux):

```
unzip oberdiek.tds.zip -d ~/texmf
```

**Script installation.** Check the directory `TDS:scripts/oberdiek/` for scripts that need further installation steps. Package `attachfile2` comes with the Perl script `pdfatfi.pl` that should be installed in such a way that it can be called as `pdfatfi`. Example (linux):

```
chmod +x scripts/oberdiek/pdfatfi.pl
cp scripts/oberdiek/pdfatfi.pl /usr/local/bin/
```

## 5.3 Package installation

**Unpacking.** The `.dtx` file is a self-extracting `docstrip` archive. The files are extracted by running the `.dtx` through plain  $\TeX$ :

```
tex kvsetkeys.dtx
```

**TDS.** Now the different files must be moved into the different directories in your installation TDS tree (also known as `texmf` tree):

```
kvsetkeys.sty          → tex/generic/oberdiek/kvsetkeys.sty
kvsetkeys.pdf          → doc/latex/oberdiek/kvsetkeys.pdf
kvsetkeys-example.tex  → doc/latex/oberdiek/kvsetkeys-example.tex
test/kvsetkeys-test1.tex → doc/latex/oberdiek/test/kvsetkeys-test1.tex
test/kvsetkeys-test2.tex → doc/latex/oberdiek/test/kvsetkeys-test2.tex
test/kvsetkeys-test3.tex → doc/latex/oberdiek/test/kvsetkeys-test3.tex
kvsetkeys.dtx          → source/latex/oberdiek/kvsetkeys.dtx
```

If you have a `docstrip.cfg` that configures and enables `docstrip`'s TDS installing feature, then some files can already be in the right place, see the documentation of `docstrip`.

## 5.4 Refresh file name databases

If your  $\TeX$  distribution (`te $\TeX$` , `mik $\TeX$` , ...) relies on file name databases, you must refresh these. For example, `te $\TeX$`  users run `texhash` or `mktextlsr`.

## 5.5 Some details for the interested

**Attached source.** The PDF documentation on CTAN also includes the `.dtx` source file. It can be extracted by AcrobatReader 6 or higher. Another option is `pdftk`, e.g. unpack the file into the current directory:

```
pdftk kvsetkeys.pdf unpack_files output .
```

**Unpacking with L<sup>A</sup>T<sub>E</sub>X.** The .dtx chooses its action depending on the format:

**plain T<sub>E</sub>X:** Run docstrip and extract the files.

**L<sup>A</sup>T<sub>E</sub>X:** Generate the documentation.

If you insist on using L<sup>A</sup>T<sub>E</sub>X for docstrip (really, docstrip does not need L<sup>A</sup>T<sub>E</sub>X), then inform the autodetect routine about your intention:

```
latex \let\install=y\input{kvsetkeys.dtx}
```

Do not forget to quote the argument according to the demands of your shell.

**Generating the documentation.** You can use both the .dtx or the .drv to generate the documentation. The process can be configured by the configuration file `ltxdoc.cfg`. For instance, put this line into this file, if you want to have A4 as paper format:

```
\PassOptionsToClass{a4paper}{article}
```

An example follows how to generate the documentation with pdfL<sup>A</sup>T<sub>E</sub>X:

```
pdflatex kvsetkeys.dtx
makeindex -s gind.ist kvsetkeys.idx
pdflatex kvsetkeys.dtx
makeindex -s gind.ist kvsetkeys.idx
pdflatex kvsetkeys.dtx
```

## 6 References

- [1] A guide to key-value methods, Joseph Wright, second draft for TUG-Boat, 2009-03-17. <http://www.texdev.net/wp-content/uploads/2009/03/keyval.pdf>
- [2] David Carlisle: *The keyval package*; 1999/03/16 v1.13; CTAN:macros/latex/required/graphics/keyval.dtx.

## 7 History

[2006/03/06 v1.0]

- First version.

[2006/10/19 v1.1]

- Fix of `\kv@set@family@handler`.
- Example added.

[2007/09/09 v1.2]

- Using package `infwarerr` for error messages.
- Catcode section rewritten.

**[2007/09/29 v1.3]**

- Normalizing and parsing of comma separated lists added.
- `\kv@normalize` rewritten.
- Robustness increased for normalizing and parsing, e.g. for values with unmatched conditionals.
- $\varepsilon$ -TeX is used if available.
- Tests added for normalizing and parsing.

**[2009/07/19 v1.4]**

- Bug fix for `\kv@normalize`: unwanted space removed (Florent Chervet).

**[2009/07/30 v1.5]**

- Documentation addition: recommendation for Joseph Wright's review article.

**[2009/12/12 v1.6]**

- Short info shortened.

**[2009/12/22 v1.7]**

- Internal optimization (`\KVS@CommaSpace`, ..., `\KVS@EqualsSpace`).

**[2010/01/28 v1.8]**

- Compatibility to `iniTeX` added.

**[2010/03/01 v1.9]**

- Support of `\par` inside values.

**[2011/01/30 v1.10]**

- Already loaded package files are not input in plain TeX.

**[2011/03/03 v1.11]**

- `\kv@break` and `\comma@break` added.

**[2011/04/05 v1.12]**

- Error message with recovery action in help message (request by GL).

**[2011/04/07 v1.13]**

- `\kv@processor@default` supports package `babel`'s shorthands.
- `\kv@set@family@handler` with shorthand support.

## 8 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; plain numbers refer to the code lines where the entry is used.

Symbols	
<code>\#</code> .....	462
<code>\\$</code> .....	187, 190
<code>\%</code> .....	538
<code>\&amp;</code> .....	188, 191
<code>\,</code> .....	234, 235, 613, 615, 622, 626, 629, 633, 683, 685
<code>\=</code> .....	36, 282, 283, 616, 618, 623, 627, 631, 635, 686
<code>\&gt;</code> .....	38, 39, 40, 41, 42, 43
<code>\@</code> .....	463, 536
<code>\@PackageError</code> .....	445
<code>\@Test</code> ....	628, 630, 632, 634, 637, 680
<code>\@ehc</code> .....	451
<code>\@empty</code> .....	13, 727, 736, 768, 777
<code>\@endslash</code> .....	13, 16, 30
<code>\@firstofone</code> .....	471, 474
<code>\@gobble</code> .....	468, 476
<code>\@makeactive</code> .....	610, 613, 616, 629, 631, 683, 686
<code>\@makeother</code> .....	615, 618, 622, 623, 626, 627, 633, 635, 685
<code>\@onelevel@sanitize</code> .....	672, 673, 724, 732, 773
<code>\@undefined</code> .....	105
<code>\\</code> ...	37, 38, 39, 40, 41, 42, 43, 44, 537
<code>\{</code> .....	460
<code>\}</code> .....	461
<code>\~</code> .....	235, 283
<b>A</b>	
<code>\advance</code> .....	501, 509, 524, 591
<code>\aftergroup</code> .....	76
<code>\AtEndDocument</code> .....	604
<b>B</b>	
<code>\begin</code> .....	34, 35, 620, 714, 764, 798
<code>\body</code> .....	480, 484
<b>C</b>	
<code>\catcode</code> .	49, 50, 52, 53, 54, 55, 56, 57, 58, 59, 60, 80, 81, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 116, 117, 119, 120, 121, 125, 126, 127, 128, 129, 130, 131, 134, 135, 137, 138, 139, 140, 144, 146, 187, 188, 190, 191, 460, 461, 462, 463, 498, 507, 515, 519, 536, 537, 538, 611
<code>\chardef</code> .....	565
<code>\comma@break</code> .....	5, <u>392</u>
<code>\comma@entry</code> .....	383, 766
<code>\comma@list</code> .....	231, 375
<code>\comma@normalize</code> .....	5, <u>222</u> , 374
<code>\comma@parse</code> .....	5, <u>373</u> , <u>778</u>
<code>\comma@parse@normalized</code> .	5, <u>375</u> , <u>377</u>
<code>\count@</code> .....	465, 494, 498, 500, 501, 505, 507, 508, 509, 513, 515, 518, 519, 523, 524
<code>\countdef</code> .....	465
<code>\csname</code> 61, 68, 97, 113, 123, 161, 164, 174, 179, 181, 397, 407, 410, 413, 420, 423, 427, 433, 436, 464, 467, 470, 473, 528, 555, 573	
<b>D</b>	
<code>\define@key</code> .....	29
<code>\dimexpr</code> .....	581
<code>\documentclass</code> .....	2, 561
<code>\dots</code> .....	38, 41, 43
<b>E</b>	
<code>\empty</code> .....	64, 65
<code>\end</code> ....	45, 46, 556, 712, 762, 796, 799
<code>\endcsname</code> .....	61, 68, 97, 113, 123, 161, 164, 174, 179, 181, 397, 405, 407, 408, 410, 413, 418, 420, 423, 427, 433, 436, 464, 467, 470, 473, 528, 555, 573
<code>\endinput</code> .....	76, 159
<code>\endlinechar</code> ....	51, 82, 118, 124, 136
<code>\endqstest</code> .....	595, 600
<code>\errmessage</code> .....	517, 614, 617
<code>\etex@unexpanded</code> .....	197, 358
<code>\Expect</code> .....	639, 642, 716, 720, 738, 740, 766, 779, 781
<b>I</b>	
<code>\ifcase</code> .....	178
<code>\ifcsname</code> .....	405, 408, 418
<code>\ifetex@unexpanded</code> .....	178
<code>\iffalse</code> ..	710, 711, 758, 759, 791, 792
<code>\ifincname</code> .....	398, 434
<code>\ifnum</code> .....	500, 508, 515, 523
<code>\iftrue</code> .....	398, 434
<code>\ifx</code> .....	62, 65, 68, 97, 105, 108, 161, 164, 174, 179, 181, 202, 291, 407, 410, 417, 420, 464, 467, 470, 473, 528, 573, 642, 719, 727, 768
<code>\immediate</code> .....	70, 99
<code>\IncludeTests</code> .....	570
<code>\input</code> .....	165, 529
<code>\iterate</code> .....	481, 483, 485
<b>K</b>	
<code>\kill</code> .....	36
<code>\kv@break</code> .....	4, <u>370</u>
<code>\kv@error@generic</code> ....	439, 442, <u>444</u>
<code>\kv@error@novalue</code> .....	421, <u>438</u>
<code>\kv@error@unknownkey</code> .....	411, <u>441</u>
<code>\kv@key</code> .....	349, 716
<code>\kv@list</code> .....	220, 331, 641, 642, 672





<code>\textgreater</code>	16	723, 726, 728, 730, 767, 769, 771
<code>\textless</code>	16	<code>\toksdef</code> . . . . . 175
<code>\texttt</code>	15	<code>\typeout</code> . . . . . 580
<code>\the</code>	16, 24, 124, 125, 126, 127, 128, 129, 130, 131, 144, 200, 218, 229, 239, 242, 250, 257, 261, 268, 272, 278, 287, 290, 299, 308, 315, 319, 326, 361, 498, 518, 519, 625, 718, 720, 723, 728, 730, 740, 769, 771, 781	
<code>\TimeDescription</code>	585, 588, 592	
<code>\TMP@EnsureCode</code>	141, 148, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158	
<code>\TMP@RequirePackage</code>	162, 168, 169	
<code>\toks@</code>	12, 16, 23, 24, 175, 199, 200, 210, 218, 224, 229, 238, 239, 242, 250, 254, 256, 257, 261, 265, 267, 268, 272, 275, 278, 286, 287, 290, 297, 299, 308, 312, 314, 315, 319, 323, 325, 326, 360, 361, 624, 625, 717, 718, 720,	
		<b>U</b>
		<code>\UNDEFINED</code> . . . . . 563
		<code>\unexpanded</code> . . . . . 562, 563, 568
		<code>\unless</code> . . . . . 405, 408, 418
		<code>\usepackage</code> . . . . . 3, 4, 5, 567, 569
		<b>V</b>
		<code>\Value</code> . . . . . 721, 723, 724, 728, 730
		<b>W</b>
		<code>\wd</code> . . . . . 740, 781
		<code>\write</code> . . . . . 70, 99
		<b>X</b>
		<code>\x</code> . . . . . 61, 62, 65, 69, 73, 75, 98, 103, 113, 122, 134, 718
		<b>Z</b>
		<code>\z@</code> . . . . . 578