# The csquotes Package

**Context Sensitive Quotation Facilities**

**Philipp Lehman, Joseph Wright**
**joseph.wright@morningstar2.co.uk**

**Version v5.2d**
**2018/04/13**

## Contents

**List of Tables** 1

**1 Introduction** 1

**2 Package Options** 2

**3 Basic Interface** 5

**4 Active Quotes** 9

**5 Integrated Interface** 12

**6 Display Environments** 14

**7 Auxiliary Commands** 15

**8 Configuration** 16

**9 Usage Notes** 26

**10 Hints and Caveats** 31

**11 Author Interface** 35

**12 Revision History** 40

## List of Tables

1 Language Options . . . . . 3
2 Styles and Variants . . . . . 17
3 Language Aliases . . . . . 18
4 Configurable Parameters . . . . 20
5 Auxiliary Hooks . . . . . 21
6 Quotation Marks . . . . . 31

## 1 Introduction

### 1.1 About csquotes

This package provides advanced facilities for inline and display quotations. It is designed for a wide range of tasks ranging from the most simple applications to the more complex demands of formal quotations. The facilities include commands, environments, and user-definable 'smart quotes' which dynamically adjust to their context. Quotation marks are switched automatically if quotations are nested and can adjust to the current language. There are additional features designed to cope with the more specific demands of academic writing. All quote styles as well as the optional active quotes are freely configurable.

### 1.2 License



### 1.3 Contributions

The multilingual support of this package depends on user contributions. If you want to contribute a quote style, please refer to § 8.1 for an overview of the components a quote style is composed of and to table 6 for a list of commands which should be used in the definition of quote styles.

## 2 Package Options

All package options are given in ⟨*key*⟩=⟨*value*⟩ syntax.

### 2.1 Main Options

`strict`=`true`, `false` default: `false`

This option turns all package warnings into error messages. This is useful to ensure that no problem will go unnoticed when finalizing a document. The short form `strict` is equivalent to `strict=true`.

`style`=⟨*style*⟩ default: `american`

This option selects a fixed quotation style. The style is used throughout the document unless it is changed manually, see § 3.7 for details. This option implicitly sets `autostyle=false`. Please refer to tables 2 and 3 for a list of available quote styles and aliases. See §§ 8.1, 8.2, 9.1 for instructions on adding or modifying quote styles.

`autostyle`=`true`, `false`, `try`, `once`, `tryonce` default: `tryonce`

This option controls multilingual support. It requires either the `babel` package or the `polyglossia` package.[2] `autostyle=true` continuously adapts the quote style to the current document language; `once` will only adapt the style once so that it matches the main language of the document. `autostyle=try` and `tryonce` are similar to `true` and `once` if multilingual support is available but will not issue any warnings

[1] http://www.latex-project.org/lppl/

[2] Note that `polyglossia` support is currently in a preliminary state because `polyglossia` is lacking a proper interface for other packages. In practice, this means that `csquotes` can detect the language (e. g., `english`) but not the language variant (e. g., `british`).

| Option key | Possible values |
|---|---|
| austrian | quotes, guillemets |
| croatian | quotes, guillemets, guillemets* |
| czech | quotes, guillemets |
| danish | quotes, guillemets, topquotes |
| english | american, british |
| french | quotes, quotes*, guillemets, guillemets* |
| galician | quotes, guillemets |
| german | quotes, guillemets, swiss |
| italian | guillemets, quotes |
| latvian | |
| norwegian | guillemets, quotes |
| portuguese | portuguese, brazilian |
| spanish | spanish, mexican |
| swedish | quotes, guillemets, guillemets* |

Table 1: Language Options Defined by Default

if not (i. e., if neither `babel` nor `polyglossia` have been loaded). The short form `autostyle` is equivalent to `autostyle=true`. See also § 3.7.

⟨*language*⟩=⟨*variant*⟩

Use the language options listed in table 1 to choose a style variant if there is more than one for a certain language. The first variant in the list is the default for the respective style. In the following example, the `autostyle` option causes `csquotes` to continuously adapt the quote style to the current language using the default style for that language. The defaults for German and Norwegian are changed:

```
\usepackage[english,ngerman]{babel}
\usepackage[autostyle,german=guillemets,norwegian=quotes]{csquotes}
```

Note that `babel`'s language name is `ngerman` here whereas `csquotes` uses `german`. When selecting a quote style automatically, this package will essentially normalize the language names first, using a list of aliases which map languages to quote styles. See table 3 for details. Table 1 indicates the language options defined by default. Additional options may be defined in the configuration file. See §§ 8.3 and 9.1 for details. Also see § 10.9 for some additional notes concerning the predefined styles.

`maxlevel`=⟨*integer*⟩ default: 2

This option controls the maximum nesting level of quotations. By default, this package supports two levels of quotations, outer and inner ones, and issues an error if quotations are nested more deeply. Alternatively, it can reuse the outer and inner quotes on further quotation levels. This alternative behavior is enabled implicitly when increasing the value of the `maxlevel` option. The minimum is 2.

`autopunct`=`true`, `false` default: `true`

This option controls whether the quotation commands scan ahead for trailing punctuation marks. See §§ 3.3, 3.5, 5.1, 5.3 for details. Also see `\DeclareAutoPunct` in § 8.9 and § 8.7 on how to configure and use the punctuation look-ahead feature. The short form `autopunct` is equivalent to `autopunct=true`.

`threshold`=⟨*integer*⟩ default: `3`

This option defines the number of lines or words the block quotation facilities use as a threshold when determining whether a quotation should be typeset in inline or in display mode. It corresponds to the `\SetBlockThreshold` command. See §§ 3.5 and 8.6 for further details.

`thresholdtype`=`lines`, `words` default: `lines`

This option selects the block threshold type. With `thresholdtype=lines`, the block quotation facilities will determine the number of lines required to typeset the quotation; with `thresholdtype=words`, they count the number of words in the quotation.[3] The default threshold setup is 3 lines. If you prefer something like 50 words, set `threshold=50` and `thresholdtype=words`. See § 3.5 for further details.

`parthreshold`=`true`, `false` default: `true`

This option fine-tunes the block threshold detection. If enabled, any explicit paragraph or line break in the quotation will trigger the threshold, i. e., the quotation will be typeset in display mode regardless of its length. If disabled, explicit paragraph or line breaks are applied normally if `thresholdtype=lines`, and treated as a regular word boundary if `thresholdtype=words`. The short form `parthreshold` is equivalent to `parthreshold=true`.

`splitcomp`=`true`, `false` default: `true`

This option is applicable with `thresholdtype=words` only. It fine-tunes the handling of compounds with explicit hyphens. If enabled, compounds are split up at all hyphens and the components are counted as separate words. If disabled, compounds are treated as a single word. The short form `splitcomp` is equivalent to `splitcomp=true`.

`csdisplay`=`true`, `false` default: `false`

By default, the block quotation facilities will automatically force inline quotations in footnotes, parboxes, minipages, and floats. Setting this option to `true` will permit context-sensitive switching to display quotations in these contexts, as in the text

[3] The word counting code is based on an idea by Donald Arseneau.

body. The short form `csdisplay` is equivalent to `csdisplay=true`. This option may also be set locally with `\csdisplaytrue` and `\csdisplayfalse`, respectively.

`debug`=`true`, `false` default: `false`

This option controls the debugging feature of the block quotation facilities. If enabled, all block quotation commands will output diagnostic messages to the transcript file. These messages indicate the calculated line/word count, explicit paragaph or line breaks detected in the quotation text, and the final inline/display determination. The short form `debug` is equivalent to `debug=true`.

### 2.2 Compatibility Options

`version`=⟨*version*⟩, `4.4`, `3.6`, `3.0`

This option is an attempt at making `csquotes` backwards compatible with earlier versions, at least to some extent. Currently, it supports backwards compatibility with version 4.4, which covers 3.7–4.4, version 3.6, which covers 3.1–3.6, and version 3.0. Older versions are not supported. It is possible to specify any arbitrary ⟨*version*⟩, even if it is not explicitly listed above. In this case, the package will keep increasing the ⟨*version*⟩ number until it either finds a suitable compatibility mode or hits the current version number. For example, when specifying `version=4.0`, `csquotes` will increase the version number until it hits 4.4, and activate the v4.4 compatibility mode because this is the highest version still compatible with the requested 4.0 release. This implies that `csquotes` may be invoked in a, to some extent, 'future proof' way by making the version which is current at the time of writing sticky. If future versions break backwards compatibility, they will automatically activate the best compatibility mode. If not, the `version` option will simply have no effect.

`babel`=`true`, `false`, `try`, `once`, `tryonce`

An older name of the `autostyle` option. This option is depreciated.

## 3 Basic Interface

This package supports two ways to tag quotations: built-in commands and active quotes defined in the document preamble or the configuration file. This section will introduce the basic commands, active quotes are discussed in § 4. When working with automated citations, you might also want to learn about the integrated quotation facilities presented in § 5.

## 3.1 Quoting Regular Text

The most basic command will simply enclose its argument in quotation marks:

`\enquote{`⟨*text*⟩`}`
`\enquote*{`⟨*text*⟩`}`

Like all quotation facilities, this command is context sensitive. Depending on the nesting level, it will toggle between outer and inner quotation marks with plain and nested quotations. The starred version of this command skips directly to the inner level. If multilingual support is enabled, the style of all quotation marks will be adapted to the current language.

## 3.2 Quoting Text in a Foreign Language

To facilitate typesetting quotations in a foreign language, `csquotes` provides two commands which combine `\enquote` with the language switches of the `babel` or the `polyglossia` package:

`\foreignquote{`⟨*lang*⟩`}{`⟨*text*⟩`}`
`\foreignquote*{`⟨*lang*⟩`}{`⟨*text*⟩`}`

This command combines `\enquote` with `\foreignlanguage`. It switches hyphenation patterns and enables the extra definitions provided by `babel/polyglossia` for ⟨*lang*⟩, which must be a language name known to the respective package. The quotation marks will match the language of the quoted piece of text.

`\hyphenquote{`⟨*lang*⟩`}{`⟨*text*⟩`}`
`\hyphenquote*{`⟨*lang*⟩`}{`⟨*text*⟩`}`

This command combines `\enquote` with the `hyphenrules` environment, that is, it merely switches hyphenation patterns. The quotation marks will match the language of the text surrounding the quotation.

## 3.3 Formal Quoting of Regular Text

Formal quotations are typically accompanied by a citation indicating the source of the quoted text. The following command is an extended version of `\enquote` which encloses the quoted piece of text in quotation marks and adds a citation after the quotation:

\textquote[⟨*cite*⟩][⟨*punct*⟩]{⟨*text*⟩}⟨*tpunct*⟩
\textquote*[⟨*cite*⟩][⟨*punct*⟩]{⟨*text*⟩}⟨*tpunct*⟩

The ⟨*text*⟩ may be any arbitrary piece of text to be enclosed in quotation marks. The optional arguments ⟨*cite*⟩ and ⟨*punct*⟩ specify the citation and any terminal punctuation of ⟨*text*⟩. ⟨*tpunct*⟩ denotes trailing punctuation after the command. If the `autopunct` option is enabled, the quotation commands will scan ahead for punctuation marks immediately following their last argument and can move them around if required. See § 8.7 on how to change the way these arguments are handled and § 9.2 for reasons why you may want to specify the punctuation as a separate argument. The starred version of this command skips directly to the inner quotation level. Here are some usage examples:

```
\textquote{...}
\textquote[][.]{...}
\textquote[Doe 1990, 67]{...}
\textquote[{\cite[67]{doe90}}]{...}
```

Note the use of the optional arguments in the examples above. As seen in the second example, ⟨*cite*⟩ is required whenever ⟨*punct*⟩ is used, even if it is empty. Also keep in mind that an optional argument containing square brackets must be wrapped in an additional pair of curly braces as shown in the last example. When working with automated citations, you might also want to learn about the integrated quotation facilities presented in § 5.

### 3.4 Formal Quoting of Text in a Foreign Language

There are two additional commands which combine `\textquote` with the language switches of the `babel` or the `polyglossia` package:

\foreigntextquote{⟨*lang*⟩}[⟨*cite*⟩][⟨*punct*⟩]{⟨*text*⟩}⟨*tpunct*⟩
\foreigntextquote*{⟨*lang*⟩}[⟨*cite*⟩][⟨*punct*⟩]{⟨*text*⟩}⟨*tpunct*⟩

This command combines `\textquote` with `\foreignlanguage`. Apart from the language, the arguments are handled as with `\textquote`.

\hyphentextquote{⟨*lang*⟩}[⟨*cite*⟩][⟨*punct*⟩]{⟨*text*⟩}⟨*tpunct*⟩
\hyphentextquote*{⟨*lang*⟩}[⟨*cite*⟩][⟨*punct*⟩]{⟨*text*⟩}⟨*tpunct*⟩

This command combines `\textquote` with the `hyphenrules` environment. Apart from the language, the arguments are handled as with `\textquote`.

## 3.5 Block Quoting of Regular Text

Formal requirements in academic writing frequently demand that quotations be embedded in the text if they are short but set off as a distinct and typically indented paragraph, a so-called block quotation, if they are longer than a certain number of lines or words. In the latter case no quotation marks are inserted. The following command deals with this requirement automatically:

`\blockquote`[⟨*cite*⟩][⟨*punct*⟩]{⟨*text*⟩}⟨*tpunct*⟩

This command determines the length of the ⟨*text*⟩. If the length exceeds a certain threshold, the ⟨*text*⟩ will be typeset in display mode, i. e., as a block quotation. If not, `\blockquote` will behave like `\textquote`. Depending on the `thresholdtype` option, the threshold may be based on the number of lines required to typeset the ⟨*text*⟩ or on the number of words in the ⟨*text*⟩. If the `parthreshold` option has been enabled, any explicit paragraph or line break in the ⟨*text*⟩ will trigger the threshold, i. e., it will be typeset in display mode regardless of its length. The default threshold setup is three lines with `parthreshold` enabled. The default environment used for display quotations is the `quote` environment. See §§ 2.1 and 8.6 on how to change these parameters. Note that `csquotes` will force inline quotations in footnotes, parboxes, minipages, and floats by default. Use the `csdisplay` option from § 2.1 to change this behavior. The optional arguments ⟨*cite*⟩ and ⟨*punct*⟩ specify the citation and any terminal punctuation of the ⟨*text*⟩. ⟨*tpunct*⟩ denotes trailing punctuation after the command. If the `autopunct` option is enabled, the quotation commands will scan ahead for punctuation marks immediately following their last argument and can move them around if required. See § 8.7 on how to change the way these arguments are handled and § 9.2 for reasons why you may want to specify the punctuation as a separate argument.

## 3.6 Block Quoting of Text in a Foreign Language

The following commands combine `\blockquote` with the language switches of the `babel` or the `polyglossia` package:

`\foreignblockquote`{⟨*lang*⟩}[⟨*cite*⟩][⟨*punct*⟩]{⟨*text*⟩}⟨*tpunct*⟩

This command behaves like `\foreignquote` if the quotation is short. If it exceeds the threshold, it will be wrapped in an `otherlanguage*` environment which is in turn wrapped in a block quotation environment. The arguments are handled as with `\blockquote`.

\hyphenblockquote{⟨*lang*⟩}[⟨*cite*⟩][⟨*punct*⟩]{⟨*text*⟩}⟨*tpunct*⟩

This command behaves like \hyphenquote if the quotation is short. If it exceeds the threshold, it will be wrapped in a hyphenrules environment which is in turn wrapped in a block quotation environment. The arguments are handled as with \blockquote.

\hybridblockquote{⟨*lang*⟩}[⟨*cite*⟩][⟨*punct*⟩]{⟨*text*⟩}⟨*tpunct*⟩

This command behaves like \hyphenquote if the quotation is short. If it exceeds the threshold, the command behaves like \foreignblockquote. In other words, it combines features of \foreignblockquote and \hyphenblockquote. The arguments are handled as with \blockquote.

### 3.7 Selecting Quote Styles

Quote styles may be selected manually at any point in the document body by way of the following command:

\setquotestyle[⟨*variant*⟩]{⟨*style*⟩}
\setquotestyle{⟨*alias*⟩}
\setquotestyle*

The regular form of this command selects a quote style and disables multilingual support. Its mandatory argument may be a quote style or an alias. If it is a quote style, the optional argument indicates the style variant. The starred version, which takes no arguments, enables multilingual support. Please refer to tables 2 and 3 for a list of available styles, style variants, and language aliases.

## 4 Active Quotes

This package also supports active characters corresponding to the commands presented in § 3. Roughly speaking, an active character is a single character which functions as a command. Like the corresponding control sequences, active quotes are fully-fledged markup elements which verify the nesting level and issue an error if quotations are nested in an invalid way. If multilingual support is enabled, the style of all quotation marks will be adapted to the current language. The commands presented in the following allocate such active quotes. They may be used in the configuration file, the preamble, or the document body. Note that all characters are automatically checked for validity as they are allocated. This package will reject characters which are unsuitable as active quotes. See § 10.3 for details on the characters which may be used as active quotes.

## 4.1 Quoting Regular Text

`\MakeOuterQuote` and `\MakeInnerQuote` define active quotes which print outer and inner quotation marks. Both take one mandatory argument, the character serving as both opening and closing mark:

`\MakeOuterQuote`{⟨*character*⟩}
`\MakeInnerQuote`{⟨*character*⟩}

`\MakeAutoQuote` defines active quotes which toggle between outer and inner quotations automatically. The two mandatory arguments serve as opening and closing mark and must be distinct:

`\MakeAutoQuote`{⟨*character 1*⟩}{⟨*character 2*⟩}
`\MakeAutoQuote*`{⟨*character 1*⟩}{⟨*character 2*⟩}

All active quotes defined with `\MakeAutoQuote` work like `\enquote`. Those defined with `\MakeOuterQuote` and `\MakeInnerQuote` cover only a part of this functionality. The former correspond to the outer level of `\enquote` whereas the latter correspond to the starred version. `\MakeAutoQuote*` is similar to `\MakeInnerQuote`, i.e. it corresponds to `\enquote*`.

## 4.2 Quoting Text in a Foreign Language

These commands combine `\MakeAutoQuote` with the language switches of the `babel` or the `polyglossia` package:

`\MakeForeignQuote`{⟨*lang*⟩}{⟨*character 1*⟩}{⟨*character 2*⟩}
`\MakeForeignQuote*`{⟨*lang*⟩}{⟨*character 1*⟩}{⟨*character 2*⟩}

The active quotes defined with the above commands are similar in concept and function to `\foreignquote` and `\foreignquote*`, respectively.

`\MakeHyphenQuote`{⟨*lang*⟩}{⟨*character 1*⟩}{⟨*character 2*⟩}
`\MakeHyphenQuote*`{⟨*lang*⟩}{⟨*character 1*⟩}{⟨*character 2*⟩}

The active quotes defined with the above commands are similar in concept and function to `\hyphenquote` and `\hyphenquote*`, respectively.

## 4.3 Block Quoting of Regular Text

`\MakeBlockQuote` defines active quotes which will set quotations inline or as a separate paragraph, depending on their length. This command takes three mandatory arguments which must be distinct:

\MakeBlockQuote{⟨*character 1*⟩}{⟨*delimiter*⟩}{⟨*character 2*⟩}

The arguments are checked for validity, see § 10.3 for details. All active quotes defined with \MakeBlockQuote behave essentially the same as \blockquote, but the handling of the citation is slightly different. ⟨*character 1*⟩ will serve as the opening mark in the source file, ⟨*character 2*⟩ as the closing one. The character indicated by the middle argument ⟨*delimiter*⟩ will serve as a delimiter separating the quoted text from the citation which is given last as the active quotes are used:

```
\MakeBlockQuote{<}{|}{>}
...
<text|citation>
```

If the delimiter is omitted, the entire text between the opening and the closing mark will be treated as quotation text.

## 4.4 Block Quoting of Text in a Foreign Language

These commands combine \MakeBlockQuote with the language switches of the babel or the polyglossia package:

\MakeForeignBlockQuote{⟨*lang*⟩}{⟨*character 1*⟩}{⟨*delimiter*⟩}{⟨*character 2*⟩}

The active quotes defined with this command are similar in concept and function to \foreignblockquote. The behavior of the delimiter character is similar to \MakeBlockQuote.

\MakeHyphenBlockQuote{⟨*lang*⟩}{⟨*character 1*⟩}{⟨*delimiter*⟩}{⟨*character 2*⟩}

The active quotes defined with this command are similar in concept and function to \hyphenblockquote. The behavior of the delimiter character is similar to \MakeBlockQuote.

\MakeHybridBlockQuote{⟨*lang*⟩}{⟨*character 1*⟩}{⟨*delimiter*⟩}{⟨*character 2*⟩}

The active quotes defined with this command are similar in concept and function to \hybridblockquote. The behavior of the delimiter character is similar to \MakeBlockQuote.

## 4.5 Controlling Active Quotes

The commands introduced above merely allocate active quotes, but these characters are not immediately made active. All allocated quotes are automatically enabled at the beginning of the document body. If any active quotes are allocated in the document body, they need to be enabled with \EnableQuotes. The following commands control the state of the active quotes within a local scope.

\EnableQuotes Enables all active quotes by redefining the allocated characters and making them active. It also restores them when disabled, set to verbatim, or overwritten.

\DisableQuotes Disables all active quotes by restoring the original category codes and definitions of all allocated characters.

\VerbatimQuotes Switches to verbatim active quotes. All active quotes will be printed verbatim until their default behavior is restored with `\EnableQuotes`.

\DeleteQuotes Disables and deallocates all active quotes, i.e. performs a complete reset of all allocated characters so that they may be newly defined.

# 5 Integrated Interface

The commands presented in this section are extended versions of some of those discussed in § 3. They differ from their counterparts in that they integrate automated citations into their syntax. Instead of adding `\cite` manually, you pass the citation arguments to the respective quotation command. See § 8.6 on how to use a command other than `\cite` to handle the citations.

## 5.1 Formal Quoting of Regular Text

The basic integrated command is an extended version of `\textquote`:

`\textcquote`[⟨*prenote*⟩][⟨*postnote*⟩]{⟨*key*⟩}[⟨*punct*⟩]{⟨*text*⟩}⟨*tpunct*⟩
`\textcquote*`[⟨*prenote*⟩][⟨*postnote*⟩]{⟨*key*⟩}[⟨*punct*⟩]{⟨*text*⟩}⟨*tpunct*⟩

The ⟨*text*⟩ may be any arbitrary piece of text to be enclosed in quotation marks. The optional arguments ⟨*cite*⟩ and ⟨*punct*⟩ specify the citation and any terminal punctuation of ⟨*text*⟩. ⟨*tpunct*⟩ denotes trailing punctuation after the command. If the `autopunct` option is enabled, the quotation commands will scan ahead for punctuation marks immediately following their last argument and can move them around if required. See § 8.7 on how to change the way these arguments are handled and § 9.2 for reasons why you may want to specify the punctuation as a separate argument. The starred version of this command skips directly to the inner quotation level. The remaining arguments are handed over to `\cite`. Note that `\cite` normally supports one optional argument only. ⟨*prenote*⟩ is only available in conjunction with the `natbib`, `jurabib`, and `biblatex` packages. How these arguments are handled depends on the citation command. With `natbib` and `biblatex`, ⟨*prenote*⟩ is in fact a notice such as 'see'. With jurabib, this argument has a different function by default. The argument ⟨*postnote*⟩, which is always available, indicates the citation postnote. This is usually a page number. ⟨*key*⟩ is the citation key. See §§ 8.6 and 8.7 on how to customize the citation.

## 5.2 Formal Quoting of Text in a Foreign Language

The following commands combine `\textcquote` with the language switches of the `babel` or the `polyglossia` package:

`\foreigntextcquote`{⟨*lang*⟩}[⟨*prenote*⟩][⟨*postnote*⟩]{⟨*key*⟩}[⟨*punct*⟩]{⟨*text*⟩}⟨*tpunct*⟩
`\foreigntextcquote*`{⟨*lang*⟩}[⟨*prenote*⟩][⟨*postnote*⟩]{⟨*key*⟩}[⟨*punct*⟩]{⟨*text*⟩}⟨*tpunct*⟩

This command combines `\textcquote` with `\foreignlanguage`. The handling of the arguments is similar to `\textcquote`.

`\hyphentextcquote`{⟨*lang*⟩}[⟨*prenote*⟩][⟨*postnote*⟩]{⟨*key*⟩}[⟨*punct*⟩]{⟨*text*⟩}⟨*tpunct*⟩
`\hyphentextcquote*`{⟨*lang*⟩}[⟨*prenote*⟩][⟨*postnote*⟩]{⟨*key*⟩}[⟨*punct*⟩]{⟨*text*⟩}⟨*tpunct*⟩

This command combines `\textcquote` with the `hyphenrules` environment. The handling of the arguments is similar to `\textcquote`.

## 5.3 Block Quoting of Regular Text

Block quotations may be combined with automated citations by using the extended version of `\blockquote`:

`\blockcquote`[⟨*prenote*⟩][⟨*postnote*⟩]{⟨*key*⟩}[⟨*punct*⟩]{⟨*text*⟩}⟨*tpunct*⟩

The difference between `\blockcquote` and `\blockquote` is that there are three citation arguments instead of one. The handling of these citation arguments is similar to `\textcquote`; see § 5.1 for details. Also see §§ 8.6, 8.7, 9.2 on how to customize block quotations.

## 5.4 Block Quoting of Text in a Foreign Language

The following commands combine `\blockcquote` with the language switches of the `babel` or the `polyglossia` package:

`\foreignblockcquote`{⟨*lang*⟩}[⟨*prenote*⟩][⟨*postnote*⟩]{⟨*key*⟩}[⟨*punct*⟩]{⟨*text*⟩}⟨*tpunct*⟩

This command combines `\blockcquote` with `\foreignlanguage`. Long quotations will be wrapped in an `otherlanguage*` environment. The handling of the citation arguments is similar to `\textcquote`.

`\hyphenblockcquote`{⟨*lang*⟩}[⟨*prenote*⟩][⟨*postnote*⟩]{⟨*key*⟩}[⟨*punct*⟩]{⟨*text*⟩}⟨*tpunct*⟩

This command combines `\blockcquote` with the `hyphenrules` environment. The handling of the citation arguments is similar to `\textcquote`.

\hybridblockcquote{⟨*lang*⟩}[⟨*prenote*⟩][⟨*postnote*⟩]{⟨*key*⟩}[⟨*punct*⟩]{⟨*text*⟩}⟨*tpunct*⟩

This command behaves like `\hyphenblockcquote` if the quotation is short, and like `\foreignblockquote` if it is long. The handling of the citation arguments is similar to `\textcquote`.

# 6 Display Environments

The environments introduced in this section will typeset quotations as a separate paragraph which looks exactly like a long quotation set by means of the block quotation facilities. Use them for quotations which are to be presented as a separate paragraph regardless of their length. Note that these environments are not replacements for the standard `quote` environment in the strict sense. They function as an additional layer on top of the latter, just like the block quotation facilities. The advantage of using these environments instead of resorting to the standard `quote` environment is that they are configurable, support citations, and will always be in sync with the block quotation facilities with respect to the configuration options discussed in §§ 8.6 and 8.7.

## 6.1 Basic Display Environments

The arguments of all display environments are generally appended to the `\begin` section of the environment:

\begin{displayquote}[⟨*cite*⟩][⟨*punct*⟩]
  \end{displayquote}

The optional arguments ⟨*cite*⟩ and ⟨*punct*⟩ specify the citation and any terminal punctuation of the quotation. See §§ 8.6 and 8.7 on how to customize this environment. Also see § 8.7 on how to change the way the optional arguments are handled and § 9.2 for reasons why you may want to specify the punctuation as a separate argument. Trailing white space at the end of the environment is removed automatically. There are two additional environments which combine `displayquote` with the language switches of the `babel` or the `polyglossia` package:

\begin{foreigndisplayquote}{⟨*lang*⟩}[⟨*cite*⟩][⟨*punct*⟩]
  \end{foreigndisplayquote}

This environment combines `displayquote` with `otherlanguage*`. Apart from the language, the arguments are handled as with `displayquote`.

`\begin{hyphendisplayquote}{⟨lang⟩}[⟨cite⟩][⟨punct⟩]`

`\end{hyphendisplayquote}`

This environment combines `displayquote` with `hyphenrules`. Apart from the language, the arguments are handled as with `displayquote`.

### 6.2 Integrated Display Environments

The following environment is an extended version of `displayquote`:

`\begin{displaycquote}[⟨prenote⟩][⟨postnote⟩]{⟨key⟩}[⟨punct⟩]`

`\end{displaycquote}`

The difference between `displaycquote` and its more basic counterpart is that there are three citation arguments instead of one. The placement of the citation is similar to `displayquote`. The handling of the citation arguments is similar to `\textcquote`, see § 5.1 for details. See §§ 8.6 and 8.7 on how to customize this environment. Also see § 8.7 on how to change the way the optional arguments are handled and § 9.2 for reasons why you may want to specify the punctuation as a separate argument. There are two environments which combine `displaycquote` with the language switches of the `babel` or the `polyglossia` package:

`\begin{foreigndisplaycquote}{⟨lang⟩}[⟨prenote⟩][⟨postnote⟩]{⟨key⟩}[⟨punct⟩]`

`\end{foreigndisplaycquote}`

This environment combines `displaycquote` with `otherlanguage*`. Apart from the language, the arguments are handled as with `displaycquote`.

`\begin{hyphendisplaycquote}{⟨lang⟩}[⟨prenote⟩][⟨postnote⟩]{⟨key⟩}[⟨punct⟩]`

`\end{hyphendisplaycquote}`

This environment combines `displaycquote` with `hyphenrules`. Apart from the language, the arguments are handled as with `displaycquote`.

## 7 Auxiliary Commands

When quoting text in a formal way, any changes applied to the quoted material, such as omissions, insertions, or alterations, are typically marked as such by using square brackets or parentheses and, where appropriate, ellipses. Use the following commands to indicate such changes in formal quotations:

\textelp{}
\textelp{⟨*text*⟩}
\textelp*{⟨*text*⟩}

When used with an empty ⟨*text*⟩ argument, this command prints an ellipsis symbol to indicate the omission of material from the quoted material. When used with a non-empty argument, the ellipsis symbol is followed by the ⟨*text*⟩ enclosed in square brackets to indicate that the ⟨*text*⟩ has been added after the omitted material. The starred version reverts the order, i. e., it prints the ⟨*text*⟩ followed by an ellipsis symbol to indicate that the ⟨*text*⟩ has been added before the omitted material. In sum, there are three ways to use this command:

```
\textelp{}       = [...]
\textelp{text}   = [...] [text]
\textelp*{text}  = [text] [...]
```

The insertion of text or individual letters may be indicated with the following command:

\textins{⟨*text*⟩}
\textins*{⟨*text*⟩}

By default, \textins will enclose the ⟨*text*⟩ added to the quoted material in square brackets. The starred version is intended for minor changes, such as the capitalization of a word, which are required to adapt the quoted material to the new context in which it is quoted.

```
\textins{text}   = [text]
\textins*{T}ext  = [T]ext
```

The deletion of individual letters may be indicated with the following command:

\textdel{⟨*text*⟩}

By default, \textdel will output two square brackets. The omitted ⟨*text*⟩ is not output.

```
text\textdel{s}  = text[]
```

See § 8.10 on how to configure the appearance of ellipses and insertions.

# 8 Configuration

If available, this package will load the configuration file csquotes.cfg. You may use this file to define new quote styles and aliases or redefine existing ones.

| Quote style | Style variants |
|---|---|
| austrian | quotes, guillemets |
| croatian | quotes, guillemets, guillemets* |
| czech | quotes, guillemets |
| danish | quotes, guillemets |
| dutch | – |
| english | american, british |
| finnish | – |
| french | quotes, quotes*, guillemets, guillemets* |
| german | quotes, guillemets, swiss |
| greek | – |
| italian | guillemets, quotes |
| norwegian | guillemets, quotes |
| portuguese | portuguese, brazilian |
| russian | – |
| spanish | spanish, mexican |
| swedish | quotes, guillemets, guillemets* |

Table 2: Quote Styles and Style Variants Defined by Default

## 8.1 Defining Quote Styles

Use the following command to define quote styles and style variants:

`\DeclareQuoteStyle`[⟨*variant*⟩]{⟨*style*⟩}[⟨*outer init*⟩][⟨*inner init*⟩]%
{⟨*opening outer mark*⟩}[⟨*middle outer mark*⟩]{⟨*closing outer mark*⟩}[⟨*kern*⟩]%
{⟨*opening inner mark*⟩}[⟨*middle inner mark*⟩]{⟨*closing inner mark*⟩}

This command may be used in the configuration file or in the document preamble. The term 'outer' refers to the first quotation level, 'inner' means quotations within another quotation. A 'middle mark' is a quotation mark inserted at the beginning of every paragraph within a quotation spanning multiple paragraphs. In most cases, the arguments defining the quotation marks will simply contain one of the commands listed in table 6. If both an outer and an inner quotation begin or end simultaneously, the kerning specified by the value ⟨*kern*⟩ will be inserted between the adjoining quotation marks. While this value can be given in any unit known to TeX, it is advisable to use the relative, font-dependent unit 'em' instead of absolute units such as points, inches, or millimeters. Note that ⟨*kern*⟩ is used as a fallback value only. If the font provides kerning data for the respective pair of quotation marks the font's kerning takes precedence. ⟨*outer init*⟩ and ⟨*inner init*⟩ are all-purpose hooks initializing the quote style. Selecting a quote style will make these hooks available to all quotation commands without expanding them. The execution of ⟨*outer init*⟩ will take place immediately before the opening outer quote is inserted, but inside the group formed by the quotation. ⟨*inner init*⟩ is executed before the opening inner

| Alias | Backend style or alias | Alias | Backend style or alias |
|---|---|---|---|
| american | english/american | newzealand | english/british |
| australian | english/british | ngerman | german |
| austrian | austrian/quotes | norsk | norwegian |
| brazil | brazilian | norwegian | norwegian/guillemets |
| brazilian | portuguese/brazilian | nswissgerman | swissgerman |
| british | english/british | nynorsk | norwegian |
| canadian | english/american | portuges | portuguese |
| croatian | croatian/quotes | portuguese | portuguese/portuguese |
| danish | danish/quotes | spanish | spanish/spanish |
| english | english/american | swedish | swedish/quotes |
| french | french/quotes | swiss | german/swiss |
| german | german/quotes | swissgerman | german/swiss |
| italian | italian/guillemets | UKenglish | british |
| mexican | spanish/mexican | USenglish | american |
| naustrian | austrian | | |

Table 3: Language Aliases Defined by Default

quote is inserted. It is advisable to avoid any global assignments in this context to prevent interference with other styles. Whenever ⟨*inner init*⟩ is used ⟨*outer init*⟩ has to be given as well, even if the argument is empty. Refer to table 2 for a list of all predefined quote styles and their variants. These are the backend styles only, see also table 3 for a list of language aliases. See § 9.1 for some examples as well as an illustration of how quote styles, aliases, and package options interact.

## 8.2 Defining Quote Aliases

The following command defines quote aliases:

`\DeclareQuoteAlias`[⟨*variant*⟩]{⟨*style*⟩}{⟨*alias*⟩}
`\DeclareQuoteAlias`{⟨*first-level alias*⟩}{⟨*second-level alias*⟩}

This command may be used in the configuration file or in the document preamble. The alias may point to a backend style or to another alias. Most language aliases refer to a backend style, but some point to an intermediate alias instead. If the alias is defined for the sake of the `babel` or the `polyglossia` package, its name must be identical to the language name used by `babel`/`polyglossia`, i. e., the expansion of `\languagename`. See § 9.1 for an illustration of how quote styles, aliases, and package options interact. A list of all aliases defined by default is given in table 3.

## 8.3 Defining Package Options

The following command creates a new package option based on a key/value syntax. It takes one mandatory argument, the quote style name:

`\DeclareQuoteOption{`⟨*style*⟩`}`

When using the new option, the name of the quote style will serve as the key. The value may be any style variant defined for the respective style. The package option will select a variant by defining an alias pointing to the desired backend style. This command is available in the configuration file only. See § 9.1 for an illustration of how quote styles, aliases, and package options interact.

## 8.4 Executing Package Options

Apart from passing options to this package as it is loaded, you may also execute options using the following command:

`\ExecuteQuoteOptions{`⟨*key=value, . . .* ⟩`}`

This command permits presetting package options in the configuration file. It may also be used in the document preamble.

## 8.5 Defining Quotes for PDF Strings

The following command specifies the quotation marks for PDF strings:

`\DeclarePlainStyle{`⟨*opening outer mark*⟩`}{`⟨*closing outer mark*⟩`}%`
`{`⟨*opening inner mark*⟩`}{`⟨*closing inner mark*⟩`}`

This command may be used in the configuration file or in the document preamble. By default, outer quotations get straight double quotes and inner quotations straight single quotes. See § 10.6 for additional hints concerning PDF strings.

## 8.6 Configuring Quotations and Citations

The following commands change the default values used by various quotation facilities of this package. The commands affected by these parameters are indicated in table 4.

`\SetBlockThreshold{`⟨*integer*⟩`}`
`\SetBlockEnvironment{`⟨*environment*⟩`}`
`\SetCiteCommand{`⟨*command*⟩`}`

`\SetBlockThreshold` defines the number of lines or words the block quotation facilities use as a threshold when determining whether a quotation should be typeset in

| Parameter | Command or environment | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | \enquote | \foreignquote | \hyphenquote | \textquote | \foreigntextquote | \hyphentextquote | \textcquote | \foreigntextcquote | \hyphentextcquote | \blockquote | \foreignblockquote | \hyphenblockquote | \blockcquote | \foreignblockcquote | \hyphenblockcquote | displayquote | foreigndisplayquote | hyphendisplayquote | displaycquote | foreigndisplaycquote | hyphendisplaycquote |
| Threshold | – | – | – | – | – | – | – | – | – | • | • | • | • | • | • | – | – | – | – | – | – |
| Environment | – | – | – | – | – | – | – | – | – | • | • | • | • | • | • | • | • | • | • | • | • |
| Cite command | – | – | – | – | – | – | • | • | • | – | – | – | • | • | • | – | – | – | • | • | • |

Table 4: Scope of Configurable Parameters

inline or in display mode. The default is three. `\SetBlockEnvironment` specifies the environment used for block and display quotations. It takes the name of an existing environment as its argument. The default is the `quote` environment provided by most document classes. The argument to `\SetCiteCommand` specifies a replacement for `\cite` which will be used by all integrated quotation facilities to handle citations. It must be a single command which takes one or two optional arguments followed by a mandatory one, the citation key. The default is `\cite`. The citation commands of the `natbib`, `jurabib`, and `biblatex` packages, which take two optional arguments, are supported.

## 8.7 Hooks for Quotations and Citations

The appearance of quotes may be configured at a low level by redefining the hooks introduced below. This section will give an overview of their syntax. See § 9.2 for practical examples. The quotation facilities which are responsive to these hooks are indicated in table 5. Also see § 8.8 for tests which may be useful when redefining the hooks.

`\mkcitation{`⟨*cite*⟩`}`

All facilities which take a ⟨*cite*⟩ argument will pass it to the `\mkcitation` hook, which may be redefined to format the citation. `\mkcitation` will only be executed if there is a citation. The default behavior is to separate the citation from the preceding text by an interword space and enclose it in parentheses. This is equivalent to the following definition:

```
\newcommand*{\mkcitation}[1]{␣(#1)}
```

| Hook | Command or environment | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | \enquote | \foreignquote | \hyphenquote | \textquote | \foreigntextquote | \hyphentextquote | \textcquote | \foreigntextcquote | \hyphentextcquote | \blockquote | \foreignblockquote | \hyphenblockquote | \blockcquote | \foreignblockcquote | \hyphenblockcquote | displayquote | foreigndisplayquote | hyphendisplayquote | displaycquote | foreigndisplaycquote | hyphendisplaycquote |
| \mktextquote | – | – | – | • | • | • | • | • | • | • | • | • | • | • | • | – | – | – | – | – | – |
| \mkblockquote | – | – | – | – | – | – | – | – | – | • | • | • | • | • | • | – | – | – | – | – | – |
| \mkbegdispquote | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – | • | • | • | • | • | • |
| \mkenddispquote | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – | • | • | • | • | • | • |
| \mkcitation | – | – | – | • | • | • | – | – | – | • | • | • | – | – | – | • | • | • | – | – | – |
| \mkccitation | – | – | – | – | – | – | • | • | • | – | – | – | • | • | • | – | – | – | • | • | • |

Table 5: Availability of Auxiliary Hooks

\mkccitation{⟨*cite code*⟩}

The integrated quotation facilities use \mkccitation instead of \mkcitation. The default behavior of this command is to separate the citation from the preceding text by an interword space. This is equivalent to the following definition:

```
\newcommand*{\mkccitation}[1]{␣#1}
```

\mktextquote{⟨*open*⟩}{⟨*text*⟩}{⟨*close*⟩}{⟨*punct*⟩}{⟨*tpunct*⟩}{⟨*cite*⟩}

The \mktextquote hook controls the layout of all text quotations. This hook is used by \textquote and related commands from §§ 3.3, 3.4, 5.1, 5.2. \blockquote and related commands from §§ 3.5, 3.6, 5.3, 5.4 use this hook for short quotations. It takes six arguments which may be arranged according to the desired output:

#1 The opening quotation mark.

#2 The ⟨*text*⟩ argument of the command.

#3 The closing quotation mark.

#4 The optional ⟨*punct*⟩ argument of the command. If there is no ⟨*punct*⟩ argument, this parameter is empty.

#5 Trailing ⟨*tpunct*⟩ punctuation immediately after the command. If there is no such punctuation or if the autopunct feature is disabled, this parameter is empty.

`#6` The optional ⟨*cite*⟩ argument of the command, wrapped in `\mkcitation`. If there is no ⟨*cite*⟩ argument, this parameter is empty. With integrated quotation commands, this parameter is the citation code, wrapped in `\mkccitation`.

By default, `\mktextquote` encloses the ⟨*punct*⟩ argument in the quotation marks along with the ⟨*text*⟩ and inserts the ⟨*cite*⟩ argument or the citation code before any trailing ⟨*tpunct*⟩ punctuation. This is equivalent to the following definition:

```
\newcommand{\mktextquote}[6]{#1#2#4#3#6#5}
```

The way in which `\mktextquote` hooks into the formatting process is best seen when looking at an example. The commands

```
\textquote[cite]{short quote}
\textcquote[55]{key1}[.]{short quote}
\blockcquote[87]{key2}{short quote}.
```

would execute `\mktextquote` with the following arguments:

```
\mktextquote{open}{short quote}{close}{}{}{\mkcitation{cite}}
\mktextquote{open}{short quote}{close}{.}{}{\mkccitation{\cite[55]{key1}}}
\mktextquote{open}{short quote}{close}{}{.}{\mkccitation{\cite[87]{key2}}}
```

where `\cite` is the command selected with `\SetCiteCommand` and `open`/`close` are internal macros which print the opening and closing quotation marks. Note that these internal macros are fully-fledged markup elements with grouping and nesting control. They must be placed in the correct order, otherwise `csquotes` will report errors about unbalanced groups or invalidly nested quotations. Since the ⟨*text*⟩ should obviously be enclosed in the quotation marks, the parameter order `#1#2#3` is effectively fixed. The parameters `#4`, `#5`, `#6` may be placed freely.

`\mkblockquote`{⟨*text*⟩}{⟨*punct*⟩}{⟨*tpunct*⟩}{⟨*cite*⟩}

The `\mkblockquote` hook controls the layout of all block quotations. This hook is used by `\blockquote` and related commands from §§ 3.5, 3.6, 5.3, 5.4 for long quotations. It takes four arguments which may be arranged according to the desired output:

`#1` The ⟨*text*⟩ argument of the command.

`#2` The optional ⟨*punct*⟩ argument of the command. If there is no ⟨*punct*⟩ argument, this parameter is empty.

`#3` Trailing ⟨*tpunct*⟩ punctuation immediately after the command. If there is no such punctuation or if the `autopunct` feature is disabled, this parameter is empty.

`#4` The optional ⟨*cite*⟩ argument of the command, wrapped in `\mkcitation`. If there is no ⟨*cite*⟩ argument, this parameter is empty. With integrated quotation commands, this parameter is the citation code, wrapped in `\mkccitation`.

By default, `\mkblockquote` inserts the ⟨*cite*⟩ argument or the citation code immediately after the ⟨*text*⟩ and adds any trailing ⟨*tpunct*⟩ punctuation at the very end. This is equivalent to the following definition:

```
\newcommand{\mkblockquote}[4]{#1#2#4#3}
```

`\mkbegdispquote`{⟨*punct*⟩}{⟨*cite*⟩}
`\mkenddispquote`{⟨*punct*⟩}{⟨*cite*⟩}

The `\mkbegdispquote` and `\mkenddispquote` hooks are used by `displayquote` and related environments from §§ 6.1 and 6.2. These hooks take two arguments:

`#1` The ⟨*punct*⟩ argument passed to the `\begin` line of the environment. If there is no ⟨*punct*⟩ argument, this parameter is empty.

`#2` The ⟨*cite*⟩ argument passed to the environment, wrapped in `\mkcitation`. If there is no ⟨*cite*⟩ argument, this parameter is empty. With integrated quotation environments, this parameter is the citation code, wrapped in `\mkccitation`.

By default, `\mkenddispquote` adds the ⟨*punct*⟩ argument as well as the ⟨*cite*⟩ argument or the citation code at the very end of the quotation. `\mkbegdispquote` does not insert anything be default. This is equivalent to the following definition:

```
\newcommand{\mkbegdispquote}[2]{}
\newcommand{\mkenddispquote}[2]{#1#2}
```

See § 9.2 for practical examples.

## 8.8 Additional Tests in Quotation Hooks

The commands in this section increase the flexibility of the hooks discussed in § 8.7. For example, it may be desirable to adjust the format of a citation depending on the way the corresponding quotation is typeset. It may also be useful to known if the quotation ends with a punctuation mark.

`\ifpunctmark`{⟨*character*⟩}{⟨*true*⟩}{⟨*false*⟩}

Expands to ⟨*true*⟩ if preceeded by the punctuation mark ⟨*character*⟩, and to ⟨*false*⟩ otherwise. The ⟨*character*⟩ may be a period, a comma, a semicolon, a colon, an exclamation mark, or a question mark. Note that this test is only available in the definition of the hooks from § 8.7.

`\ifpunct{`⟨*true*⟩`}{`⟨*false*⟩`}`

Expands to ⟨*true*⟩ if preceeded by any punctuation mark, and to ⟨*false*⟩ otherwise. Note that this test is only available in the definition of the hooks from § 8.7.

`\ifterm{`⟨*true*⟩`}{`⟨*false*⟩`}`

Expands to ⟨*true*⟩ if preceeded by a terminal punctuation mark (period, exclamation mark, or question mark), and to ⟨*false*⟩ otherwise. Note that this test is only available in the definition of the hooks from § 8.7.

`\iftextpunctmark{`⟨*text*⟩`}{`⟨*character*⟩`}{`⟨*true*⟩`}{`⟨*false*⟩`}`

Executes ⟨*true*⟩ if the ⟨*text*⟩ ends with the punctuation mark ⟨*character*⟩, and to ⟨*false*⟩ otherwise. The ⟨*character*⟩ may be a period, a comma, a semicolon, a colon, an exclamation mark, or a question mark. This command is robust.

`\iftextpunct{`⟨*text*⟩`}{`⟨*true*⟩`}{`⟨*false*⟩`}`

Executes ⟨*true*⟩ if the ⟨*text*⟩ ends with any punctuation mark, and to ⟨*false*⟩ otherwise. This command is robust.

`\iftextterm{`⟨*text*⟩`}{`⟨*true*⟩`}{`⟨*false*⟩`}`

Executes ⟨*true*⟩ if the ⟨*text*⟩ ends with a terminal punctuation mark (period, exclamation mark, or question mark), and to ⟨*false*⟩ otherwise. This command is robust.

`\ifblockquote{`⟨*true*⟩`}{`⟨*false*⟩`}`

Expands to ⟨*true*⟩ in all block and display quotations, and to ⟨*false*⟩ otherwise.

`\ifblank{`⟨*string*⟩`}{`⟨*true*⟩`}{`⟨*false*⟩`}`

This generic command, which is provided by the `etoolbox` package, expands to ⟨*true*⟩ if the ⟨*string*⟩ is blank (empty or spaces), and to ⟨*false*⟩ otherwise. This is useful to test for an empty argument in the definition of the `\mk...quote` commands. Note that this test is redundant in the definition of the citation hooks because they are only executed if there is a citation.

`\unspace` Removes preceding whitespace, i. e., removes all skips and penalties from the end of the current horizontal list.

## 8.9 Configuring Punctuation Look-Ahead

`\DeclareAutoPunct{`⟨*characters*⟩`}`

This command defines the punctuation marks to be considered by the quotation commands as they scan ahead for punctuation. Note that ⟨*characters*⟩ is an undelimited list of characters. Valid ⟨*characters*⟩ are period, comma, semicolon, colon, exclamation and question mark. The default setting is:

```
\DeclareAutoPunct{.,;:!?}
```

This definition is restored automatically whenever the `autopunct` package option is set to `true`. Executing `\DeclareAutoPunctuation{}` is equivalent to setting `autopunct=false`, i. e., it disables this feature.

## 8.10 Configuring Auxiliary Commands

The appearance of ellipses and insertions formatted with the auxiliary commands from § 7 is controlled by six hooks. When `\textelp` is used with an empty argument (ellipsis only), it will execute `\mktextelp`. When used with a non-empty ⟨*text*⟩ argument (ellipsis and insertion), the ⟨*text*⟩ will be passed as an argument to `\mktextelpins`. The starred form will pass the ⟨*text*⟩ to `\mktextinselp` instead. These are the default definitions:

```
\newcommand{\mktextelp}{[\textellipsis\unkern]}
\newcommand{\mktextelpins}[1]{[\textellipsis\unkern]␣[#1]}
\newcommand{\mktextinselp}[1]{[#1]␣[\textellipsis\unkern]}
```

The `\textins` command passes its ⟨*text*⟩ argument to `\mktextins` for further processing. The starred variant of `\textins` uses `\mktextmod` instead. These are the default definitions:

```
\newcommand{\mktextins}[1]{[#1]}
\newcommand{\mktextmod}[1]{[#1]}
```

The `\textdel` command passes its ⟨*text*⟩ argument to `\mktextdel` for further processing. This is the default definition (note that the argument is not output):

```
\newcommand{\mktextdel}[1]{[]}
```

You may redefine the above hooks to change the format of the printed output. For example, if you prefer replacements to be indicated by "[. . . text]" rather than "[. . .] [text]", redefine `\mktextelpins` accordingly:

```
\newcommand{\mktextelpins}[1]{[\textellipsis #1]}
```

The \unkern in the default definitions is required because \textellipsis adds asymmetric kerning by default. The kerning after the final dot is similar to the spacing between the dots, which is fine if \textellipsis is followed by any text, but undesirable if it is enclosed in brackets.

# 9 Usage Notes

## 9.1 Adding a New Quote Style

This section will give some comprehensive examples of how to define new quote styles. The examples presented here will only make use of the most basic components a quote style can be composed of. The main point is to illustrate the interaction of quote styles, variants, aliases, and package options. To get started, consider a simple house style which may be selected by way of the package option style or the command \setquotestyle:

```
\DeclareQuoteStyle{house}
  {\textquotedblleft}{\textquotedblright}
  {\textquoteleft}{\textquoteright}
```

Now suppose that we wanted to add a quote style for an imaginary language called Newspeak and that there were two quote styles commonly used in Newspeak, an official one and an unofficial one. In this case, we need two backend styles implemented as variants of the newspeak style, newspeak/official and newspeak/unofficial:

```
\DeclareQuoteStyle[official]{newspeak}
  {\textquotedblleft}{\textquotedblright}
  {\textquoteleft}{\textquoteright}
\DeclareQuoteStyle[unofficial]{newspeak}
  {\textquotedblright}{\textquotedblleft}
  {\textquoteright}{\textquoteleft}
```

The official variant should be the default for this style. There is no need to copy the definition of the official variant to accomplish that. We simply define an alias labeled newspeak which points to the desired variant:

```
\DeclareQuoteAlias[official]{newspeak}{newspeak}
```

The reason why we are using variants and aliases instead of two independent styles will become clear in a moment. Suppose that the babel package offered support for Newspeak, but this language was known to babel as otherspeak:

```
\DeclareQuoteAlias{newspeak}{otherspeak}
```

This is an example of a second-level alias pointing to a first-level alias. If the current language is `otherspeak`, the above aliases will be expanded as follows:

```
otherspeak = newspeak = newspeak/official
```

We also define a new package option to choose a style variant:

```
\DeclareQuoteOption{newspeak}
```

This will add a new package option with a key called `newspeak`. The value of this option may be any variant of the `newspeak` style defined in the configuration file. In this example, there are two possible values: `official` and `unofficial`. To select the default or the alternative style for the entire document we use:

```
\usepackage[style=newspeak]{csquotes}
\usepackage[style=newspeak,newspeak=unofficial]{csquotes}
```

To select the default or the alternative style with multilingual support we use:

```
\usepackage[babel]{csquotes}
\usepackage[babel,newspeak=unofficial]{csquotes}
```

The base style must be implemented as an alias in this case since the `newspeak` option will select a variant by redefining and thus overwriting the `newspeak` alias. Since the `otherspeak` alias points to `newspeak` and not directly to any backend style, using the `newspeak` option will also have the desired effect if multilingual support is enabled. Note that there are some style names which have a special meaning. See § 10.9 for details.

## 9.2 Using Quotation and Citation Hooks

Style guides for writers usually make detailed provisions concerning the formatting of quotations and citations, including rules dealing with punctuation placement. This section will discuss some typical usage scenarios, using hooks and other facilities introduced in §§ 8.7, 8.8, 8.9. In the examples below, we assume the following input:

```
\textquote[citation][.]{This is a complete sentence}
\textquote[citation][]{This is an incomplete sentence}.
```

We start off with semantically strict punctuation placement, i. e., terminal punctuation is enclosed in the quotation marks if it in fact is part of the quotation, and printed after the closing mark if it is not. Our first sample cases will use German quotation marks, but the formatting convention, as far as punctuation placement is concerned, is very common. We assume citations in footnotes in the first example, hence the desired output is as follows:

„This is a complete sentence.“[1]

„This is an incomplete sentence“.[2]

This is accomplished by the following definitions:

```
\renewcommand{\mkcitation}[1]{\footnote{#1}}
\renewcommand{\mktextquote}[6]{#1#2#4#3#5#6}
```

In some cases, slightly different placement of the punctuation in the second line is requested:

„This is a complete sentence.“[1]

„This is an incomplete sentence“[2].

This is accomplished by the following definitions:

```
\renewcommand{\mkcitation}[1]{\footnote{#1}}
\renewcommand{\mktextquote}[6]{#1#2#4#3#6#5}
```

Let's switch to American quotation marks in the next examples. When using parenthetical citations, the terminal punctuation mark is typically placed at the very end of the entire sentence, after the closing quotation mark and the citation, even if it is part of the original quotation:

“This is a complete sentence” (citation).

“This is an incomplete sentence” (citation).

This is accomplished by the following definitions:

```
\renewcommand{\mkcitation}[1]{ (#1)}
\renewcommand{\mktextquote}[6]{#1#2#3#6#4#5}
```

The American quotation style is special in that it requires that a period or a comma immediately after a closing quotation mark be moved inside the quotes, even if it is not part of the original quotation. Given the above input (and assuming citations in footnotes in the next example), we need the same output in both cases:

“This is a complete sentence.”[1]

“This is an incomplete sentence.”[2]

This is accomplished by the following definitions:

```
\DeclareAutoPunct{.,}
\renewcommand{\mkcitation}[1]{\footnote{#1}}
\renewcommand{\mktextquote}[6]{#1#2#4#5#3#6}
```

The style usually seen in French books uses semantically strict punctuation placement. The unusual aspect of this style is the footnote mark, which is placed inside the quotes, before the terminal punctuation mark. Given the above input, we need the following output:

« This is a complete sentence[1]. »

« This is an incomplete sentence[2] ».

This is accomplished by the following definitions:

```
\renewcommand{\mkcitation}[1]{\footnote{#1}}
\renewcommand{\mktextquote}[6]{#1#2#6#4#3#5}
```

Note that the spacing of the quotation marks, as is common in French typography, is handled by the quote style, as defined with `\DeclareQuoteStyle`. There is no need to deal with these details when redefining `\mktextquote` and similar hooks. `\mktextquote` will get the quotation marks plus spacing as parameters `#1` and `#3`.

In addition to language and style specific adaptions, the formatting hooks discussed in this section may be used to further automate the quoting process. For example, they can be configured to automatically insert an ellipsis mark when quoting a sentence truncated at the end. In the following examples, we assume these lines of input:

```
\textquote[citation]{This is an incomplete sentence}.
\textquote[citation][.]{This is a complete sentence}
\textquote[citation]{This is a complete sentence.}
```

Let's assume American-style quotes combined with citations in footnotes. Our previous definition for that was:

```
\renewcommand{\mktextquote}[6]{#1#2#4#5#3#6}
```

We need to insert an ellipsis mark if the ⟨*punct*⟩ argument of the citation command is empty, or if it was omitted. Parameter `#4` will be blank in both cases:

```
\renewcommand{\mktextquote}[6]{%
  #1#2\ifblank{#4}{ \textelp{}}{#4}#5#3#6}
```

Given the above input, this definition will yield the following output:

"This is an incomplete sentence [...]."[1]

"This is a complete sentence."[2]

"This is a complete sentence. [...]"[3]

The first two cases are handled correctly but the third one needs more tuning: there is a spurious ellipsis because the final period is not passed to `\textquote` as a separate ⟨*punct*⟩ argument, but included in the quotation. To fix that, we add an additional `\ifpunct` test to check if the quoted text ends with a punctuation mark, and omit the ellipsis if this is the case:

```
\renewcommand{\mktextquote}[6]{%
  #1#2\ifblank{#4}{\ifpunct{}{ \textelp{}}}{#4}#5#3#6}
```

The `\ifpunct` test will check the last character in `#2` and omit the ellipsis when detecting a punctuation mark. This yields the desired output:

"This is an incomplete sentence [...]."[1]

"This is a complete sentence."[2]

"This is a complete sentence."[3]

When using automated citations, it is convenient to employ the integrated quotations commands from § 5. For example, instead of this:

```
\textquote[{\cite[55]{key}}][.]{This is a complete sentence}
\textquote[{\cite[55]{key}}][]{This is an incomplete sentence}.
```

you might use `\textcquote` instead of `\textquote`:

```
\textcquote[55]{key}[.]{This is a complete sentence}
\textcquote[55]{key}{This is an incomplete sentence}.
```

The definition of the punctuation hooks is the same in both cases, but citations are set up in a slightly different way in the second case, using `\mkccitation` instead of `\mkcitation`. The first line in the example below is intended for parenthetical citations, the second one for citations in footnotes:

```
\renewcommand{\mkccitation}[1]{ (#1)}
\renewcommand{\mkccitation}[1]{\footnote{#1}}
```

Advanced citation packages such as `natbib` and `biblatex` provide dedicated commands for various types of citations. In this case, it is advisable to hand control of citations to the respective citation command, i. e., `\mkccitation` is used to add spacing where required but will leave the rest to the citation command. In the case of `biblatex`, this boils down to:

```
\SetCiteCommand{\parencite}
\renewcommand{\mkccitation}[1]{ #1}
```

for parenthetical citations and

```
\SetCiteCommand{\footcite}
\renewcommand{\mkccitation}[1]{#1}
```

for citations in footnotes.

| Double quotation marks | | Single quotation marks | |
|---|---|---|---|
| **Command** | **Example** | **Command** | **Example** |
| \textquotedblleft | “AaGg“ | \textquoteleft | ‘AaGg‘ |
| \textquotedblright | ”AaGg” | \textquoteright | ’AaGg’ |
| \quotedblbase | „AaGg„ | \quotesinglbase | ‚AaGg‚ |
| \guillemotleft | «AaGg« | \guilsinglleft | ‹AaGg‹ |
| \guillemotright | »AaGg» | \guilsinglright | ›AaGg› |

Table 6: Quotation Marks Provided by Encodings T1, LY1, ETU

# 10 Hints and Caveats

## 10.1 Input Encodings

The active quotes provided by this package may depend on or benefit from the `inputenc`/`inputenx` packages under certain circumstances. As long as the active quotes are in the range 0–127, there is no benefit in loading `inputenc`. If you are using an 8-bit input encoding such as `latin1`, `inputenc` is required for the quotes to function properly in a *verbatim* context. It should therefore be loaded before any active quotes are allocated (not necessarily before `csquotes` is loaded). The macro-level UTF-8 support of this package builds on the `utf8` module of the inputenc package. When using this encoding, make sure that inputenc is loaded with the `utf8` option. Do not use the `utf8x` option as this would implicitly load the `ucs` package which is not supported by `csquotes`. UTF-8 encoding will be detected automatically. All commands discussed in § 4 work as usual with this encoding. XeTeX and LuaTeX in native UTF-8 mode will also work as expected and do not require any additional packages. See also § 10.5.

## 10.2 Output Encodings

The OT1 font encoding, which is the default output encoding of LaTeX, merely includes the quotation marks used in English. You will need an encoding like T1, LY1, or, with XeTeX or LuaTeX, TU, in order to get guillemets or baseline quotation marks. This package deliberately refrains from providing any workarounds for the OT1 legacy encoding. If you need T1 or some other extended encoding for some of the quotation marks, you will most likely need it anyway to get proper hyphenation for the respective language. See table 6 for a list of common quotation marks included in T1, LY1, and TU.

## 10.3 Valid Active Quotes

In general, an active quote may be any single character with category code 12 or 13, or any multibyte UTF-8 sequence representing a single character. However, there are a few exceptions: numbers, punctuation marks, the apostrophe, the hyphen, and all characters which are part of the LaTeX syntax are rejected. In sum, the following characters will be considered as reserved by this package: `A-Z a-z 0-9 . , ; : ! ? ' - # $ % & ^ _ ` ~ \ @ * { } [ ]`

## 10.4 Invalid Nesting and Unbalanced Active Quotes

Every quotation forms a group which includes both the quoted piece of text and the quotation marks. This package tracks the nesting level of all quotations and thus allows for basic validation. If quotations are nested in an invalid way, it will issue an error message. Keep in mind that the active quotes are more than a convenient way to enter quotation marks. They are fully-fledged markup elements which imply grouping as well, hence they must always be balanced and must not interfere with other group boundaries. This package ensures that an error is triggered if quotes are unbalanced or nested in an invalid way. However, note that packages generally can not catch low-level errors caused by grouping mistakes, nor do they have any control over the wording of generic error messages.

## 10.5 Active Quotes in Special Contexts

All quotation commands are designed for use in text mode and will issue an error message in math mode. Note that all active quotes retain their original function in math mode. It is perfectly possible to use a character like the greater-than symbol as an active quote without interfering with math mode. In a verbatim context, the active quotes will normally be disabled. If a character is in the range 128–255, its original function is restored so that the `inputenc/inputenx` package may handle it in verbatim environments. This feature is available with the standard verbatim environments and with those provided by (or defined via) the packages `verbatim`, `fancyvrb`, `moreverb`, and `alltt`. This also applies to the `\verb` command and the `shortvrb` package. The `listings` package provides dedicated support for extended input encodings. When using this package, activate its 'extended characters' option and specify the input encoding. Note that `listings` does not support macro-level UTF-8 decoding. XeTeX and LuaTeX in native UTF-8 mode will work as expected and do not require the `inputenc` package.

Some care is still required when choosing active quotes. Note that you normally cannot use active characters in the argument to commands expecting a string of characters, such as `\input`, `\label`, or `\cite`. There are two packages which try to remedy this situation: the `babel` package and the `underscore` package (when

loaded with the strings option). Both packages redefine several standard commands affected by this general problem. If any one of these packages is loaded, csquotes will take advantage of all improvements automatically. Unfortunately, both packages patch a different set of commands and neither one covers all possibly vulnerable commands.

## 10.6 PDF Strings and hyperref Support

This package interfaces with the hyperref package as PDF strings such as bookmarks are generated. See § 8.5 on how to configure the quotation marks used in PDF strings. Support for PDF strings is only available with the basic facilities presented in §§ 3.1 and 3.2 as well as §§ 4.1 and 4.2. Be advised that the way hyperref builds PDF strings imposes severe limitations on the capabilities of all commands. Most notably, the nesting level of quotations cannot be tracked in this context. Nested quotations will generally get outer marks, but you may use starred commands or active inner quotes to request inner marks explicitly. If quotation marks are to be included in the document properties of a PDF file, you must use \hypersetup to specify the strings. The replacement mechanism will not function within the optional argument to \usepackage. See the hyperref manual for further details.

## 10.7 Footnotes Inside Quotations

This package will automatically reset the nesting level within any footnote included in a quotation. If the babel or the polyglossia package has been loaded, it will also reset the language. The language of the footnote text including the hyphenation patterns will match the language of the text surrounding the quotation. This applies to parboxes, minipages, and floats as well.

## 10.8 Using csquotes with babel's Shorthands

The commands discussed in § 11.5 may be combined with the shorthands of the babel package such that babel provides the user interface and csquotes the backend.[4] For example, the German module of the babel package defines, amongst other things, the shorthands "‘ and "’. Such shorthands are input aids, i. e., physical markup elements with a fixed definition. Typing "‘ is a short way of saying \quotedblbase but it is not different in concept. These shorthands can be transformed into ‘smart quotes’ which behave like \enquote. Here is a simple ad hoc solution suitable for documents with only one language:

[4] This also applies to the polyglossia package if the babelshorthands option of polyglossia has been enabled.

```
\documentclass{...}
\usepackage[german]{babel}
\usepackage[babel=once]{csquotes}
\defineshorthand{"`}{\openautoquote}
\defineshorthand{"'}{\closeautoquote}
```

It is possible to move such definitions to `csquotes.cfg`. In this case, the code is slightly more complex because it needs to be more general:

```
\AtEndPreamble{%
  \@ifpackageloaded{babel}
    {\iflanguage{german}
       {\declare@shorthand{german}{"`}{\openautoquote}%
        \declare@shorthand{german}{"'}{\closeautoquote}}
       {}}
    {}}
```

This code redefines the shorthands only if the `babel` package has been loaded and the main language is `german`. Note that `babel`'s shorthands are language-specific. The way they are configured and handled is technically and conceptually different from the active quotes discussed in § 4. Active quotes are defined globally and automatically adapt to the current language. With `babel`, each language has its own set of shorthands. Also note that `babel` uses `\AtBeginDocument` to initialize the main document language, including the corresponding shorthands. We use `\AtEndPreamble` to defer the code until the end of the preamble. This way, we can be sure that `babel` has been loaded but that the main document language has not been initialized yet. See the `babel` manual for further details. The `\AtEndPreamble` command is provided by the `etoolbox` package.

## 10.9 Miscellaneous Notes about the Predefined Styles

There are three styles which serve a special purpose: `default`, `fallback`, and `debug`. The `default` style is a dynamic alias pointing to the default quote style used if the multilingual interface is not enabled. The package option `style` and the command `\setquotestyle` will redefine this alias. The `fallback` style is a backend style used as a fallback whenever the multilingual interface is enabled but there is no matching quote style for the current language. It will print bold question marks by default. The `debug` style will not print quotation marks but the current quote level as a bold number. This style may be selected using the `style` option or the `\setquotestyle` command. It is intended for debugging only.

All variants of the `french` style use spaced out guillemets as outer marks. The style variant `quotes` uses double quotes as inner marks. The starred variant `quotes*` is similar to its regular counterpart except that it will also space out the inner marks.

The `guillemets` variant employs spaced out guillemets on all levels. It will also insert guillemets at the beginning of every paragraph inside a quotation spanning multiple paragraphs. In addition to that, two adjoining marks at the end of a quotation are replaced by a single one; if two nested quotations end simultaneously, the second closing mark is omitted automatically. The starred variant `guillemets*` is similar to its regular counterpart, differing only in the middle mark inserted at the beginning of every paragraph. The regular variant uses a left-pointing guillemet whereas the starred one uses a right-pointing one.

# 11 Author Interface

The following sections discuss the programmer interface to the `csquotes` package as well as some details of the implementation. They are intended for class and package authors who want to interface with this package.

## 11.1 Controlling Active Quotes

The author commands in this section behave essentially like the corresponding user commands discussed in § 4.5. The only difference is that they work quietly behind the scenes without writing any notices to the transcript file. The scope of these commands is local so that all changes may be confined to a group. Note that the active quotes are enabled at the beginning of the document body. Under no circumstances will this package make any characters active in the document preamble. You will only need the following commands when dealing with active quotes at the beginning of or in the document body.

`\@enablequotes` This command enables all characters allocated as active quotes. It also restores their definitions if they were disabled or accidentally overwritten. With single-byte encodings, this command (re)defines all allocated characters and makes them active. With UTF-8 encoding, it redefines the internal macro used by the `inputenc` package to typeset the respective UTF-8 sequence (`\u8:`⟨*character*⟩). UTF-8 characters in the range 0–127 are handled as with single-byte encodings. When using a TeX engine with native UTF-8 support, such as XeTeX, all characters are handled as with single-byte encodings.

`\@disablequotes` This command restores the *status quo ante* of all active quotes. With single-byte encodings, there are two possible cases. (1) If a character had already been active when it was allocated as active quote, its former definition is restored. (2) If a character had not been active when it was allocated, its former category code is restored. With UTF-8 encoding, this command restores the former definition of the internal macro used by the `inputenc` package to typeset the respective UTF-8 sequence. UTF-8 characters in the range 0–127 are handled as with single-byte

encodings. When using a TeX engine with native UTF-8 support, such as XeTeX, all characters are handled as with single-byte encodings.

\@verbatimquotes For verbatim environments and similar applications, use this command rather than \@disablequotes. It redefines the active quotes in a way that is better suited for verbatim typesetting. With single-byte encodings, it will do one of the following things. (1) If a character is in the range 0–127, it is redefined such that it expands to itself with category code 12. (2) If a character is in the range 128–255, there are two possibilities. (a) If it had already been active when it was allocated, its former definition is restored. (b) If it had not been active before, it is redefined such that it expands to itself with its former category code.

Characters in the range 0–127 are added to the \dospecials list. Characters in the range 128–255 remain active, permitting the inputenc package to typeset them verbatim (due to case 2a, which implies that you must load inputenc before allocating active quotes). Case 2b is usually undesirable in verbatim environments. If inputenc is loaded, however, this should not happen. With UTF-8 encoding, this command restores the former definition of the internal macro used by the inputenc package to typeset the respective UTF-8 sequence. UTF-8 characters in the range 0–127 are handled as with single-byte encodings. When using a TeX engine with native UTF-8 support, such as XeTeX, all characters are handled as with single-byte encodings.

Due to case 1, \@verbatimquotes itself is independent of any \dospecials processing. You may typeset all active quotes verbatim by using this command exclusively. The advantage of this approach is that it does not require any category code changes, hence this command may also be used to modify an argument after it has been read. Also note that the standard LaTeX verbatim environments as well as all environments provided by or defined via the packages verbatim, fancyvrb, moreverb, and alltt are catered for automatically. This also applies to the \verb command and the shortvrb package.

\@deletequotes This command implicitly executes \@disablequotes and deallocates all active quotes, which results in a complete reset of all active quotes so that they may be newly defined. This command should be used with care because the reset is not visible to the user. Using \DeleteQuotes may be preferable.

## 11.2 Active Quotes in a Strings-Only Context

A possible problem with active characters are strings-only contexts, i. e. cases in which an active character is used in the formation of a control sequence name. A typical example is the \label command which expects a string of characters. Any active character may break \label when used in its argument. There are two

packages which try to remedy this situation, albeit in different ways: `babel` and `underscore`.

The `babel` package defines the switch `\if@safe@actives` and patches several standard commands such that the switch is set to `true` while they process their arguments. The approach taken by the `underscore` package is slightly different. If `underscore` is loaded with the `strings` option, it patches several commands such that `\protect` is equivalent to `\string` while the arguments are processed. If any one of these packages is loaded, `csquotes` will take advantage of that automatically. Unfortunately, both packages patch a different set of commands and neither one covers all possibly vulnerable commands. If `babel` is loaded, for example, you may use active quotes in the argument of `\label`, but not in the argument of `\input`. If you load `underscore` with its `strings` option, active quotes may also be used in the argument of `\input`.

When writing a package which may have to process user-supplied arguments in a strings-only context, there are two ways to deal with active quotes. Taking the approach of the `babel` package, you may do the following:

```
\let\if@safe@actives\iftrue
```

This is best done in a group. If grouping is not feasible, you must ensure that the switch is properly restored. In contrast to using `\@safe@activestrue`, this approach works even if `babel` is not loaded. However, note that you must take three states into account when restoring the switch in this case: true, false, and undefined. Taking the approach of the `underscore` package, you may also do the following:

```
\let\@@protect\protect \let\protect\string
```

This could either be done in a group or without any grouping, but followed by `\restore@protect`. The first approach works with the active characters of the `babel` and the `underscore` packages. The second one works with the `underscore` and the `at` packages. Unfortunately, the active characters of the `inputenc` package support neither of the above-mentioned techniques. As far as `csquotes` is concerned, it does not matter which approach you take. In both cases all active quotes expand to themselves with category code 12. With macro-level UTF-8 support, UTF-8 encoded active quotes expand to a string of characters with category code 12. This string will be valid UTF-8. In a verbatim `\write` operation, you should employ one of the techniques discussed in this section rather than `\@verbatimquotes`, which is geared to verbatim typesetting.

## 11.3 Block Quotations

The block quotation facilities need to typeset all quotations twice. The first pass is required to measure the length of the quotation. The actual typesetting takes place

on the second pass, in a format depending on the result of the first one. In order to prevent any side-effects of the first (trial) pass, the `csquotes` package (1) performs the first pass inside a group, (2) employs checkpointing to freeze all LaTeX counters, and (3) sets `\if@filesw` to `false`. However, it can not prevent side-effects caused by commands that (1) make any global assignments which are not overwritten on the second pass (for example, by way of `\g@addto@macro`), (2) increment counters globally in a way that circumvents LaTeX's counter commands, or (3) do not check `\if@filesw` every time they are about to write to an auxiliary file. If you observe any malfunctions related to the trial pass (for example, if counters are incremented twice or if an item appears twice in a list), use `\BlockquoteDisable` to redefine or disable the affected command temporarily.

`\BlockquoteDisable{`⟨*code*⟩`}`

The ⟨*code*⟩ may be arbitrary LaTeX code which redefines vulnerable commands locally such that they work differently during the trial pass. The ⟨*code*⟩ itself should obviously not include any global assignments. This solution should be considered as a last ressort but may be the quickest way to fix a vulnerable package. Note that there is no need to escape parameter characters by doubling them in the ⟨*code*⟩ argument. Simply use this command like `\AtBeginDocument` and similar hooks.

## 11.4 Registering Quotation Marks

In order to track punctuation marks inside quotations, this package requires that all quotation marks be transparent to the space factor, i. e., that they have a space factor code of zero. This setting is specific to the output encoding. Settings for the encodings OT1, OT2, OT4, T1, LY1, LGR, T2A, T2B, T2C, LCY, X2, and ETU are provided by default. Other encodings may be set up in the configuration file using the following command:

`\DeclareQuoteGlyph{`⟨*encoding*⟩`}{`⟨*position*⟩`}`

The ⟨*encoding*⟩ is the name of the output encoding, for example `OT1`. This string corresponds to the identifiers used when loading the `fontenc` package. The ⟨*position*⟩ argument is an integer indicating the position of a glyph in this encoding. You need to reqister all quotation marks in the output encoding, using one declaration for each glyph. As an example, these are the settings for the OT1 encoding:

```
\DeclareQuoteGlyph{OT1}{34}% = \textquotedblright
\DeclareQuoteGlyph{OT1}{39}% = \textquoteright
\DeclareQuoteGlyph{OT1}{92}% = \textquotedblleft
\DeclareQuoteGlyph{OT1}{96}% = \textquoteleft
```

The ⟨*position*⟩ argument may use any notation accepted by TeX in integer assignments, e. g., `171` in decimal or `"00AB` in hexadecimal notation. See the settings in `csquotes.def` for further examples. The advantage of registering quotes with the above command (rather than adjusting the space factor codes globally) is that the declarations are only used locally inside quotations and will not affect any other part of the document.

## 11.5 Automatic Quotation Marks

The commands in this section provide access to the automatic quotation facilities at a slightly lower level than the user commands in §§ 3.1 and 4.1. In contrast to the commands discussed in § 11.6, the facilities in this section are fully-fledged markup elements which verify the nesting level and issue an error if quotations are nested in an invalid way. They form groups and must always be balanced, see § 10.4 for details. In other words, the facilities in this section are semantic markup elements, the ones in § 11.6 are physical markup elements.

`\openautoquote` Opens a nestable quotation.

`\closeautoquote` Closes a nestable quotation.

In terms of their function, the above commands correspond to the regular versions of `\enquote` and `\MakeAutoQuote`. The following commands correspond to the starred variants `\enquote*` and `\MakeAutoQuote*`:

`\openinnerquote` Opens an inner quotation.

`\closeinnerquote` Closes an inner quotation.

The above commands may be used to implement an alternative user interface. For example, you can combine them with the shorthands of the `babel` package such that `babel` provides the user interface and `csquotes` the backend. See § 10.8 and the `babel` manual for details.

## 11.6 Internal Quotation Marks

The commands in this section print the quotation marks of the current style, as defined with `\DeclareQuoteStyle`, without any grouping or nesting control. The quotation marks reflect all changes to the quotation style. If the multilingual interface is enabled, they are also synced with the current language.

`\textooquote` Prints the opening outer quotation mark of the currently active quote style.

`\textcoquote` Prints the closing outer quotation mark.

`\textmoquote` Prints the middle outer quotation mark.

`\textoiquote` Prints the opening inner quotation mark.

`\textciquote` Prints the closing inner quotation mark.

`\textmiquote` Prints the middle inner quotation mark.

Note that the initialization hooks for the respective quotation style are not executed automatically. They may be accessed separately:

`\initoquote` Executes the outer initialization hook.

`\initiquote` Executes the inner initialization hook.

The scope of these hooks must always be confined to a group.

## 12 Revision History

This revision history is a list of changes relevant to users of this package. Changes of a more technical nature which do not affect the user interface or the behavior of the package are not included in the list. If an entry in the revision history states that a feature has been *extended*, this indicates a syntactically backwards compatible modification, such as the addition of an optional argument to an existing command. Entries stating that a feature has been *modified*, *renamed*, or *removed* demand attention. They indicate a modification which may require changes to existing documents in some, hopefully rare, cases. The `version` option from § 2.1 may be helpful in this case. The numbers on the right indicate the relevant section of this manual.

### 5.2d 2018-04-13

Update for LaTeX kernel changes in 2018

### 5.2c 2018-02-11

Added language option `galician` . . . . . 2.1

Added language option `latvian` . . . . . 2.1

Improve language support for `czech`

### 5.2b 2017-03-11

Added language option `portuguese` . . . . . 2.1

### 5.2a 2017-02-03

Fix behaviour of `\blockquote` inside `\parbox` and related constructs

### 5.2 2016-12-28

Add `\textdel` auxiliary command. . . . . 7

### 5.1h 2016-07-14

Adapt `\fixligatures` to work correctly with LuaTeX

### 5.1g 2016-01-31

Update for new TU Unicode encoding

### 5.1f 2015-07-18

Update `(n)swissgerman` and `(n)austrian` quote styles

### 5.1e 2015-04-15

New maintainer: Joseph Wright

Update `danish` quote style

### 5.1d 2011-10-22

Slightly modified quote style `italian/guillemets`

Made variant `guillemets` the default for `italian`

Fixed XeTeX kerning issue

### 5.1c 2011-03-25

Fixed spurious language reset

### 5.1b 2011-01-20

Fixed conflict with `polyglossia` and French

### 5.1a 2011-01-11

Fix for paragraph environments inside `\blockquote`

### 5.1 2010-11-19

Added package option `debug` . . . . . 2.1
Added package option `threshold` . . . . . 2.1
Added package option `thresholdtype` . . . . . 2.1
Added package option `parthreshold` . . . . . 2.1
Added package option `splitcomp` . . . . . 2.1
Improved block quotation facilities . . . . . 3.5
Improved integrated block quotation facilities . . . . . 5.3

### 5.0c 2010-09-21

Added package option `csdisplay` . . . . . 2.1
Added `\csdisplaytrue` and `\csdisplayfalse` . . . . . 2.1
Fixed conflict with `polyglossia`

### 5.0b 2010-08-06

Fixed issue with `babel`'s active punctuation marks

### 5.0a 2010-06-09

Fixed bug related to middle quote marks
Minor internal update for biblatex

### 5.0 2010-06-02

Renamed package option `babel` to `autostyle` . . . . . 2.1
Added compatibility option `babel` . . . . . 2.2
Added preliminary `polyglossia` interface . . . . . 2.1
Added package option `autopunct` . . . . . 2.1
Added package option `version` . . . . . 2.2
Added `\hybridblockquote` . . . . . 3.6
Added `\hybridblockcquote` . . . . . 5.4
Added `\MakeHybridBlockQuote` . . . . . 4.4
Added `\DeclareAutoPunct` . . . . . 8.9
Extended `\textquote` . . . . . 3.3
Extended `\foreigntextquote` . . . . . 3.4
Extended `\hyphentextquote` . . . . . 3.4
Extended `\blockquote` . . . . . 3.5
Extended `\foreignblockquote` . . . . . 3.6
Extended `\hyphenblockquote` . . . . . 3.6
Extended `\textcquote` . . . . . 5.1
Extended `\foreigntextcquote` . . . . . 5.2
Extended `\hyphentextcquote` . . . . . 5.2
Extended `\blockcquote` . . . . . 5.3
Extended `\foreignblockcquote` . . . . . 5.4
Extended `\hyphenblockcquote` . . . . . 5.4

Added \mktextquote . . . . . 8.7
Added \mkblockquote . . . . . 8.7
Added \mkbegdispquote . . . . . 8.7
Added \mkenddispquote . . . . . 8.7
Removed \mkpretextpunct . . . . . 8.7
Removed \mkmidtextpunct . . . . . 8.7
Removed \mkfintextpunct . . . . . 8.7
Removed \mkpreblockpunct . . . . . 8.7
Removed \mkmidblockpunct . . . . . 8.7
Removed \mkfinblockpunct . . . . . 8.7
Removed \mkpredisppunct . . . . . 8.7
Removed \mkmiddisppunct . . . . . 8.7
Removed \mkfindisppunct . . . . . 8.7
Removed \quotetext . . . . . 8.7
Removed \quoteblock . . . . . 8.7
Renamed \ifquotepunct to \ifpunct . . . . . 8.8
Renamed \ifquoteterm to \ifterm . . . . . 8.8
Added \ifpunctmark . . . . . 8.8
Removed \ifquotecolon . . . . . 8.8
Removed \ifquotecomma . . . . . 8.8
Removed \ifquoteexclam . . . . . 8.8
Removed \ifquoteperiod . . . . . 8.8
Removed \ifquotequestion . . . . . 8.8
Removed \ifquotesemicolon . . . . . 8.8
Renamed \ifstringblank to \ifblank . . . . . 8.8
Added \unspace . . . . . 8.8
Added \textelp . . . . . 7
Added \textins . . . . . 7
Added \mktextelp . . . . . 8.10
Added \mktextelpins . . . . . 8.10
Added \mktextinselp . . . . . 8.10
Added \mktextins . . . . . 8.10

Added `\mktextmod` . . . . . 8.10
Expanded documentation . . . . . 9.2
Removed `\cquote` legacy alias (use `\textcquote` instead) . . . . . 5.1
Removed `\foreigncquote` legacy alias (use `\foreigntextcquote`) . . . . 5.2
Removed `\hyphencquote` legacy alias (use `\hyphentextcquote`) . . . . . 5.2
Removed `\RestoreQuotes` legacy alias (use `\EnableQuotes`) . . . . . 4.5
Removed `\@restorequotes` legacy alias (use `\@enablequotes`) . . . . . 11.1
Added support for EU2 encoding . . . . . 11.4
Added quote style for Croatian

### 4.4d 2010-02-06

Added quote style for Dutch
Added quote style for Finnish
Added quote style for Greek
Added support for LGR encoding . . . . . 11.4

### 4.4c 2009-09-23

Fixed incompatibility with `inputenx` package

### 4.4b 2009-07-24

Fixed bug in glyph declarations for EU1 encoding . . . . . 11.4

### 4.4a 2009-07-04

Added support for EU1 encoding . . . . . 11.4
Added quote style for Brazilian
Added preliminary quote style for Portuguese

### 4.4 2009-05-30

Added package option `maxlevel` . . . . . 2.1
Added special quote style `debug` . . . . . 10.9
Added `\DeclareQuoteGlyph` . . . . . 11.4
Added support for OT2 and OT4 encodings . . . . . 11.4
Added support for T2A, T2B, T2C, X2 encodings . . . . . 11.4
Added support for LCY encoding . . . . . 11.4
Added quote style for Russian

### 4.3 2008-11-23

Made package option `babel=tryonce` the default setting . . . . . 2.1

Internal updates for `biblatex` package

### 4.2 2008-06-24

Upgrade to `etoolbox` 1.6

### 4.1 2008-04-11

Fixed timing issue with active quotes introduced in 4.0

### 4.0 2008-03-02

e-TeX now mandatory requirement

New dependency on `etoolbox` package

Added package option `spanish` . . . . . 2.1

Added variant `mexican` to style `spanish` . . . . . 2.1

Removed variant `oldstyle` from `english` style . . . . . 2.1

Removed variant `oldstyle` from `french` style . . . . . 2.1

Removed variant `imprimerie` from `french` style . . . . . 2.1

Expanded documentation . . . . . 10.8

Added `\openautoquote` and `\closeautoquote` . . . . . 11.5

Added `\openinnerquote` and `\closeinnerquote` . . . . . 11.5

Moved predefined styles, variants, options to `csquotes.def`

Added more hints and examples to `csquotes.cfg`

Added extended PDF bookmarks to this manual

### 3.8 2008-01-05

Added variant `guillemets*` to style `swedish` . . . . . 2.1

Added language alias `australian` . . . . . 8.2

Added language alias `newzealand` . . . . . 8.2

Internal improvements

### 3.7 2007-03-25

Added package option `babel=try` . . . . . 2.1

Added package option `babel=tryonce` . . . . . 2.1

Added `\MakeAutoQuote*` . . . . . 4.1

Added `\MakeForeignQuote*` . . . . . 4.2
Added `\MakeHyphenQuote*` . . . . . 4.2
Added `\mkpretextpunct` . . . . . 8.7
Added `\mkpreblockpunct` . . . . . 8.7
Added `\mkpredisppunct` . . . . . 8.7
Removed compatibility code for `\blockcite` legacy command
Internal updates for `biblatex` package

### 3.6 2006-11-09

Added `\BlockquoteDisable` . . . . . 11.3
Fix for `amsmath` package (active quotes in `split` and other environments)
Fix for `endnotes` package (endnotes in block quotations)
Revised Spanish quote style

### 3.5 2006-08-24

Exchanged definitions of French `quotes` and `quotes*` variants . . . . . 10.9
Internal updates for `inputenc` 1.1 b (2006-05-05)

### 3.4 2006-04-02

Stricter validation of user-defined active characters . . . . . 10.3
Author interface now documented in this manual . . . . . 11
Added documentation of `\@enablequotes` . . . . . 11.1
Added documentation of `\@disablequotes` . . . . . 11.1
Added documentation of `\@verbatimquotes` . . . . . 11.1
Added documentation of `\@deletequotes` . . . . . 11.1
Added documentation concerning string handling . . . . . 11.2
Added documentation of interface to internal marks . . . . . 11.6

### 3.3 2006-02-27

Added support for UTF-8 encoded active quotes . . . . . 10.1
Modified active quotes, category codes 7, 8 no longer valid . . . . . 10.3
Modified delimiters, category codes 3, 4, 7, 8 no longer valid . . . . . 10.3
Active quotes may now be defined in the document body . . . . . 4.5
Renamed `\RestoreQuotes` to `\EnableQuotes` . . . . . 4.5

Added `\DeleteQuotes` . . . . . 4.5
Added `\VerbatimQuotes` . . . . . 4.5
Added `\ExecuteQuoteOptions` . . . . . 8.4
Added package option `babel=once` . . . . . 2.1
Added new style variant for French . . . . . 10.9
Improved nesting control of active block quotes
Made active block quotes robust

### 3.2 2005-12-05

Added quote style for Spanish
Fixed bug in `hyperref` interface

### 3.1 2005-08-29

Added `\textquote` . . . . . 3.3
Added `\foreigntextquote` . . . . . 3.4
Added `\hyphentextquote` . . . . . 3.4
Renamed `\cquote` to `\textcquote` . . . . . 5.1
Renamed `\foreigncquote` to `\foreigntextcquote` . . . . . 5.2
Renamed `\hyphencquote` to `\hyphentextcquote` . . . . . 5.2
Extended `\textcquote` . . . . . 5.1
Extended `\foreigntextcquote` . . . . . 5.2
Extended `\hyphentextcquote` . . . . . 5.2
Modified environment `displayquote` . . . . . 6.1
Modified environment `foreigndisplayquote` . . . . . 6.1
Modified environment `hyphendisplayquote` . . . . . 6.1
Extended environment `displaycquote` . . . . . 6.2
Extended environment `foreigndisplaycquote` . . . . . 6.2
Extended environment `hyphendisplaycquote` . . . . . 6.2
Added `\mkmidtextpunct` . . . . . 8.7
Added `\mkfintextpunct` . . . . . 8.7
Added `\mkmiddisppunct` . . . . . 8.7
Added `\mkfindisppunct` . . . . . 8.7
Added auxiliary environment `quotetext` . . . . . 8.7

Added detection of paragraphs to all block quotation facilities . . . . . 3.5
`\ifquote...` now usable in `\mkcitation` and `\mkccitation` . . . . . 8.8
Terminal punctuation now evaluated by all quotation facilities
Prevent undesirable `?‘` and `!‘` ligatures in T1 encoding
Always adjust space factor codes of backend quotes

### 3.0 2005-07-14

Extended `\blockquote` . . . . . 3.5
Extended `\foreignblockquote` . . . . . 3.6
Extended `\hyphenblockquote` . . . . . 3.6
Extended `\setquotestyle` . . . . . 3.7
Added `\cquote` . . . . . 5.1
Added `\foreigncquote` . . . . . 5.2
Added `\hyphencquote` . . . . . 5.2
Added `\blockcquote` . . . . . 5.3
Added `\foreignblockcquote` . . . . . 5.4
Added `\hyphenblockcquote` . . . . . 5.4
Added environment `displayquote` . . . . . 6.1
Added environment `foreigndisplayquote` . . . . . 6.1
Added environment `hyphendisplayquote` . . . . . 6.1
Added environment `displaycquote` . . . . . 6.2
Added environment `foreigndisplaycquote` . . . . . 6.2
Added environment `hyphendisplaycquote` . . . . . 6.2
Modified `\DeclarePlainStyle` . . . . . 8.5
Added `\SetCiteCommand` . . . . . 8.6
Renamed `\blockcite` to `\mkcitation` . . . . . 8.7
Added `\mkccitation` . . . . . 8.7
Added `\mkmidblockpunct` . . . . . 8.7
Added `\mkfinblockpunct` . . . . . 8.7
Added `\ifquotepunct` . . . . . 8.8
Added `\ifquoteterm` . . . . . 8.8
Added `\ifquoteperiod` . . . . . 8.8

Added \ifquotecomma . . . . . 8.8
Added \ifquotesemicolon . . . . . 8.8
Added \ifquotecolon . . . . . 8.8
Added \ifquoteexclam . . . . . 8.8
Added \ifquotequestion . . . . . 8.8
Added \ifstringblank . . . . . 8.8
Added evaluation of terminal punctuation within block quotations
With \nonfrenchspacing, adjust space factor codes of backend quotes
Improved nesting control when running under e-TeX