The *csquotes* package

Context sensitive markup for inline quotations

Philipp Lehman
plehman@gmx.net

Version 2.4
November 1, 2004

Contents

Tables

1 Introduction

1.1 Abstract

For authors in the humanities and the social sciences quotations are as crucial as code fragments for computer scientists and equations for mathematicians. While Latex provides environments for typesetting quotations as separate paragraphs, there is no markup for quotations embedded in the running text. Apart from that, you might need to type lengthy control sequences to even get proper quotation marks in some languages. Not only is that tedious, it also decreases the readability of the document source. The csquotes package addresses these issues by providing markup elements for inline quotations. It offers control sequences and optionally active characters and toggles between inner and outer quotation marks for nested quotes. It can interface with the babel package and adapt the quotation marks to the current language automatically. All quote styles as well as the optional active characters are fully configurable.

1.2 License

1.3 Contributions

Since it is impossible for an individual author to support all languages covered by the babel package, the csquotes package depends on contributions. If your language is not supported or if you spot a mistake in the default styles, contact me at the address given above. I would also be grateful for any affirmative

Option key	Possible values
babel	*none*, true, false
strict	*none*, true, false
style	*quote style or alias*
danish	quotes, guillemets
english	american, british, oldstyle
french	quotes, guillemets, guillemets*, oldstyle, imprimerie
german	quotes, guillemets, swiss
italian	quotes, guillemets
norwegian	guillemets, quotes
swedish	quotes, guillemets

Table 1: Package options defined by default

feedback. If the styles for your language are correct and the defaults correspond to common practice, let me know so that I can mark them as verified. Refer to section 6.1 for an overview of the components a quote style is composed of. Further comments concerning some of the default styles are to be found at the end of section 2.4. See table 2 for a list of control sequences used in the definition of quote styles. Please use these control sequences if you want to contribute a quote style.

1.4 Acknowledgments

I am indebted to Donald Arseneau and Mark Wooding for valuable hints. Additional thanks go to Robert Schlicht for thorough bug reports and testing.

2 Package options

The csquotes package employs a key/value syntax for all package options. Table 1 lists all option keys defined by default along with their possible values. It is possible to define additional options in the configuration file. See section 6.3 for details.

2.1 The 'babel' option

Use this option to enable babel support. If the babel interface is enabled, the style of all quotation marks will automatically be adapted to the current language. Note that babel support is not enabled by default, even if the babel package has been loaded. Instead of the verbose form `babel=true` you may also use the short form `babel`.

2.2 The 'strict' option

Use this option to turn all package warnings into error messages. The csquotes package features a certain degree of fault tolerance by default. If it encounters a problem that is not fatal, csquotes will normally issue a warning and attempt to continue. When finalizing a document, however, you might want to ensure that no problem will go unnoticed. Instead of the verbose form `strict=true` you may also use the short form `strict`.

2.3 The 'style' option

If you are not using the babel package or if you do not want csquotes to adapt the quote style to the current language, you may also select a quote style manually. The selected style will be used throughout the entire document unless it is changed explicitly (see section 3.5). The `babel` option will override any fixed style. Please refer to tables 3 and 4 for a list of all available quote styles and aliases. Adding new styles or changing the predefined ones is straightforward. See sections 6.1 and 6.6 for details.

2.4 The language options

In some languages there is more than one quote style. Use the language options listed in table 1 to choose a style variant. The first variant in the list is the default for the respective style. Let us consider some examples. In the first example, csquotes will adapt the quote style to the current language, using the default style for that language. We also change some of the defaults. In the English sections of the text, the quotation marks will adhere to the British standard; in the Swedish sections csquotes will use guillemets:

```
\usepackage[babel,english=british,swedish=guillemets]{csquotes}
```

Here is another example. The babel package does not include dedicated support for Swiss German. This dialect is covered by the `ngerman` language extension. However, the quotation marks used in the German-speaking parts of Switzerland differ from the practice in Germany and Austria. To get proper Swiss quotation marks, you need to instruct csquotes to use the `swiss` variant of the `german` style:

```
\usepackage[style=german,german=swiss]{csquotes}
```

If babel support is enabled, the options would be given as follows:

```
\usepackage[ngerman]{babel}
\usepackage[babel,german=swiss]{csquotes}
```

Note that babel's language name is `ngerman` whereas csquotes uses `german`. Essentially, csquotes will normalize babel's language names first, using a list of aliases which map languages to quote styles. These aliases are required since some languages are known to the babel package by more than one name. In our example, `ngerman` is different from `german` as far as babel is concerned but this difference is irrelevant for csquotes, hence `ngerman` is defined as a quote alias of `german` by default. Csquotes will expand all aliases first and select the quote style after that. A list of all predefined aliases is given in table 4.

A few additional notes concerning some style variants are in order. All variants of the `french` style use guillemets as outer marks and space out the marks automatically. The `quotes` variant uses double quotes as inner marks whereas the `guillemets` variant employs guillemets on all levels. The latter variant will also add a guillemet at the beginning of every new paragraph inside a quotation

spanning multiple paragraphs. In addition to that, two adjoining marks at the end of a quotation will be replaced by a single one. If an outer and an inner quotation end simultaneously, there will only be a single closing guillemet. This behavior is modelled after a feature of the frenchpro package except that csquotes does not require the use of any additional control sequences. The second closing guillemet will be 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.

The `oldstyle` variants available for some styles are also worth mentioning. They will place an opening outer mark at the beginning of every line within the quotation. Note that this feature requires Omega, a backwards compatible extension of Tex. You need to compile the Latex source with `lambda` if you want to make use of those variants. This feature will not be available if the source is compiled with `latex`. The `oldstyle` variant of the `english` style is based on the British quoting convention and will insert a single quote at the beginning of the line. The respective variant of the `french` style uses a left-pointing guillemet instead. The `imprimerie` variant of the `french` style, which is based on a directive of the Imprimerie Nationale, the French government printing office, requires Omega as well. This quote style is rather difficult to describe. Essentially, it incorporates all features of the `guillemets` variant, that is, guillemets are used on all levels, a left-pointing guillemet is inserted at the beginning of every paragraph inside an outer quotation spanning multiple paragraphs, and two adjoining marks at the end of a quotation are replaced by a single one. In addition to that, this variant will insert a left-pointing guillemet at the beginning of every line within an inner quotation spanning multiple lines.

3 Control sequences

The csquotes package provides two ways to tag quotations: built-in control sequences and custom active characters defined in the document preamble or the configuration file. This section will introduce the built-in quoting macros.

3.1 Quoting regular text

The most basic macro provided by the csquotes package will simply enclose its argument in quotation marks:

```
\enquote{#1}
\enquote*{#1}
```

This macro takes one mandatory argument, the quoted piece of text. `\enquote` is context sensitive. Depending on the nesting level, it will toggle between outer and inner quotation marks for plain quotations and quotations inside another quotation respectively. The starred version jumps to the inner level right away. If babel support is enabled by means of the respective package option, the style of all quotation marks will be adapted to the current language.

3.2 Quoting text in a foreign language

To make typesetting quotations in a foreign language more convenient, csquotes features two macros which combine `\enquote` with babel's language switches.

```
\foreignquote{#1}{#2}
\foreignquote*{#1}{#2}
```

The `\foreignquote` macro combines the features of `\enquote` with those of babel's `\foreignlanguage` macro. It takes two mandatory arguments:

`#1` babel's language name
`#2` quoted text

`\foreignlanguage` switches hyphenation patterns and enables all extra definitions for the respective language, such as shorthands and, for some languages, micro-typographical adjustments. In addition to that, `\foreignquote` will typeset quotation marks matching the language of the quoted piece of text. The starred version of this macro jumps directly to the inner quotation level.

```
\hyphenquote{#1}{#2}
\hyphenquote*{#1}{#2}
```

The `\hyphenquote` macro, which takes the same arguments, combines the features of `\enquote` with an inline version of babel's `hyphenrules` environment. Unlike `\foreignquote`, it merely changes the hyphenation patterns. It will neither enable any extra definitions provided by babel nor will it adapt the quotation marks to the language of the quoted piece of text. The quotation marks will match the language of the text surrounding the quotation instead. The starred version of this macro jumps directly to the inner quotation level.

3.3 Block quoting of regular text

It is common practice in many fields of academic publishing to set quotations inline if they are short but set them off as an indented paragraph if they are longer than a certain number of lines. In the latter case no quotation marks are inserted. The csquotes package provides a macro which may be used instead of `\enquote` to deal with this requirement automatically:

```
\blockquote[#1]{#2}{#3}
```

This macro takes one optional and two mandatory arguments:

`#1` line threshold (optional)
`#2` quoted text
`#3` text after quotation

`\blockquote` determines the number of lines required to typeset the quotation given as `#2`. If the quotation is short, `\blockquote` will act like `\enquote`. If it is longer than a given number of lines, it will be wrapped in a block quote environment instead. Here are some examples:

```
\blockquote{...}{}
\blockquote{...}{\space\cite[185]{doe90}}
\blockquote[8]{...}{\footnote{...}}
```

Argument #1 may be zero or any positive integer. Argument #3 will be inserted immediately after the closing quotation mark if the quotation is short. If the quotation exceeds the threshold, it will be inserted at the end of the quotation, but inside the block quote environment. By default, the threshold is set to five lines and the environment used to typeset quotations exceeding this threshold is the `quote` environment provided by most Latex classes. See section 6.5 on how to change these default values and further customize block quotations.

3.4 Block quoting of text in a foreign language

There are two additional macros which combine the features of `\blockquote` with babel's language switches:

```
\foreignblockquote{#1}[#2]{#3}{#4}
```

This macro takes one optional and three mandatory arguments:

#1 babel's language name
#2 line threshold (optional)
#3 quoted text
#4 text after quotation

`\foreignblockquote` works like `\foreignquote` if the quotation is short. If it exceeds the threshold, it will be wrapped in babel's `otherlanguage*` environment which is in turn wrapped in a block quote environment. There is a similar replacement for `\hyphenquote`:

```
\hyphenblockquote{#1}[#2]{#3}{#4}
```

This macro, which takes the same set of arguments, works like `\hyphenquote` for short quotations. If a quotation exceeds the threshold, it will be wrapped in babel's `hyphenrules` environment which is in turn wrapped in a block quote environment.

3.5 Switching quote styles

If the babel interface is not enabled, you may select a quote style manually by using the following macro at the desired location in the document body:

```
\setquotestyle[#1]{#2}
```

This macro takes one optional and one mandatory argument:

#1 quote style variant (optional)
#2 quote style name or alias

Please refer to tables 3 and 4 for a list of available quote styles, style variants, and language aliases.

4 Active characters

In addition to the control sequences discussed in section 3, csquotes also supports so-called active characters. An active character is a single character which works like a control sequence. Designating individual characters to serve as markup elements is very convenient with csquotes, which comes with several macros facilitating this. These macros may be used in the document preamble or in the configuration file.

4.1 Quoting regular text

`\MakeOuterQuote` and `\MakeInnerQuote` define active characters which typeset either outer or inner quotation marks respectively. Both macros take one mandatory argument, the character to serve as opening and closing quotation mark:

```
\MakeOuterQuote{#1}
\MakeInnerQuote{#1}
```

`\MakeAutoQuote` defines active characters which will toggle between inner and outer quotation marks automatically:

```
\MakeAutoQuote{#1}{#2}
```

This macro takes two mandatory arguments which must be distinct:

`#1` opening quotation mark
`#2` closing quotation mark

Here are some examples:

```
\MakeOuterQuote{"}
\MakeInnerQuote{|}
\MakeAutoQuote{<}{>}
```

The active characters defined by `\MakeAutoQuote` work exactly like `\enquote`. Those defined by `\MakeOuterQuote` and `\MakeInnerQuote` cover only a certain part of this functionality. The former macro corresponds to the outer level of `\enquote` without nesting support whereas the latter corresponds to the starred version of `\enquote`.

4.2 Quoting text in a foreign language

You can also define active characters which will typeset quotation marks and change the language which will be in effect within the quotation:

```
\MakeForeignQuote{#1}{#2}{#3}
\MakeHyphenQuote{#1}{#2}{#3}
```

Both macros take three mandatory arguments:

`#1` babel's language name
`#2` opening quotation mark
`#3` closing quotation mark

Here are some examples:

```
\MakeForeignQuote{french}{«}{»}
\MakeHyphenQuote{spanish}{<}{>}
```

The active characters defined by `\MakeForeignQuote` are similar in function to the `\foreignquote` macro whereas the ones defined by `\MakeHyphenQuote` work like `\hyphenquote`.

4.3 Restoring active characters

If the definitions of active characters set up by the macros introduced above get overwritten by another package, you can restore them with the following macro:

`\RestoreQuotes`

This macro will 'replay' the definitions of all active characters set up in the configuration file and the document preamble. It is set up so that it can only be used in the document body. Since csquotes will defer the setup of active characters until the beginning of the document body anyway, using it in the preamble would at best be redundant and might even interfere with other packages.

5 Hints and limitations

5.1 Invalid nesting and unbalanced active quotes

Every quotation forms a group which includes both the quoted piece of text and the quotation marks. The csquotes 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 a warning and typeset black boxes as error markers. Note that active quote characters 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. The csquotes package can detect unbalanced quotation marks in some cases, but it cannot intercept errors caused by complex grouping mistakes.

5.2 Active quotes in math mode and special contexts

As a matter of course, the quote macros provided by csquotes have been designed for Tex's text mode. If you inadvertently use them in math mode, csquotes will issue an error message. The package will also try to disable all active quote characters in math mode by restoring their original category codes as Tex enters math or display math mode. This should allow for using characters like the greater-than symbol or special characters such as the underscore as active quotes without interfering with math mode. If a character had already been active before it was redefined by csquotes, it will be assigned to the category 'other' in math mode. In addtion to that, csquotes will disable active quotes in `verbatim` environments and while the argument of `\verb`, `\index`, and `\glossary` is parsed by assigning them to the category 'other' in the respective context. Some care is still required when it comes to choosing active quote characters. Please

note that you cannot use active quotes in the argument of macros and primitives which expect a string of characters, such as `\cite`, `\ref`, `\input`, `\include`, `\includegraphics` and so on.

5.3 PDF strings and hyperref support

If you load the hyperref package and use any quote macros in a heading, the quotation marks will be included in the bookmarks of PDF files. Csquotes will automatically interface with the hyperref package, but the capabilities of all macros are severely limited in this context. Most notably, there is no way for csquotes to track the quote nesting level and to distinguish between opening and closing quotation marks automatically. This limitation is due to the way hyperref handles PDF strings. By default, csquotes will replace outer quotation marks with straight double quotes and inner quotation marks with straight single quotes in PDF strings. Nested quotations will also get outer marks, but you can still use starred macros like `\enquote*` or active characters set up with `\MakeInnerQuote` to specify inner quotation marks explicitly. The replacement mechanism can only work in PDF file information strings if you use `\hypersetup` to specify the strings. Quote macros or active quotation marks cannot be used in the optional argument of `\usepackage` while loading the hyperref package. For further details on the limitations in PDF strings see the hyperref documentation, most notably the files `paper.pdf` and `slides.pdf`. See also section 6.4 on how to change the characters used as quotation marks in PDF strings.

5.4 Language switching in page headers

Some of the standard Latex classes have the questionable feature of converting all page headers to uppercase strings. This behavior can interfere with babel's language switches if they are used in a chapter title which also serves as a running head. If csquotes tries to pass the string 'language' to babel in this context, babel will in fact get 'LANGUAGE' and abort with an error message complaining about an undefined language. Csquotes works around this problem by converting all language names to lowercase before passing them on to babel. This works well for almost all language names except some aliases containing capital letters. Fortunately, there are only two such aliases, namely `USenglish` and `UKenglish`. The workaround is simple: always use `american` and `british` instead.

5.5 Footnotes within quotations

The csquotes package will automatically reset the quote nesting level within any footnote included in a quotation. If babel support is enabled, it will also reset the language of the footnote. As far as babel is concerned, the language of the footnote text including the hyphenations patterns will match the language of the text surrounding the quotation.

5.6 Output encodings

The OT1 font encoding, the default output encoding of Tex, merely includes the quotation marks used in English. You will need to switch to T1 encoding in or-

Double marks		Single marks	
Control sequence	**Glyph**	**Control sequence**	**Glyph**
`\textquotedblleft`	“AaGg“	`\textquoteleft`	‘AaGg‘
`\textquotedblright`	”AaGg”	`\textquoteright`	’AaGg’
`\quotedblbase`	„AaGg„	`\quotesinglbase`	‚AaGg‚
`\guillemotleft`	«AaGg«	`\guilsinglleft`	‹AaGg‹
`\guillemotright`	»AaGg»	`\guilsinglright`	›AaGg›

Table 2: Quotation marks included in T1 encoding

der to get guillemets or baseline quotation marks, for example. The csquotes package deliberately refrains from providing any workarounds for the obsolete OT1 encoding. If you need T1 encoding for some of the quotation marks, you will most likely need it anyway to get proper hyphenation for the respective language. See table 2 for a list of quotation marks included in T1 encoding.

6 Configuration

The csquotes package will read a configuration file called `csquotes.cfg` if it is available. You may use this file to define additional quote styles and aliases or redefine existing ones.

6.1 Defining quote styles

Use `\DeclareQuoteStyle` to define new quote styles and variants or redefine existing ones:

```
\DeclareQuoteStyle[#1]{#2}[#3][#4]{#5}[#6]{#7}[#8]{#9}[#10]{#11}
```

This macro takes six optional and five mandatory arguments:

- `#1` quote style variant (optional)
- `#2` quote style name
- `#3` outer quote initialization (optional)
- `#4` inner quote initialization (optional)
- `#5` opening outer quotation mark
- `#6` middle outer quotation mark (optional)
- `#7` closing outer quotation mark
- `#8` kerning between adjoining marks (optional)
- `#9` opening inner quotation mark
- `#10` middle inner quotation mark (optional)
- `#11` closing inner quotation mark

`\DeclareQuoteStyle` 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 control sequences listed in table 2. If the middle outer and no

Quote style	Style variants
danish	quotes, guillemets
dutch	–
english	american, british, oldstyle
finnish	–
french	quotes, guillemets, guillemets*, oldstyle, imprimerie
german	quotes, guillemets, swiss
italian	quotes, guillemets
norwegian	guillemets, quotes
swedish	quotes, guillemets

Table 3: Quote styles and style variants defined by default

middle inner mark is defined, `#6` will span the entire quotation. You may set `#10` to `\relax` in this case to suppress the middle outer mark within the scope of an inner quotation. The kerning `#8` is inserted between adjoining quotation marks if both an outer and an inner quotation begin or end simultaneously. While the kerning value can be specified in any unit understood by Tex, it is advisable to use the font-dependent unit 'em'. `#3` and `#4` are all-purpose hooks initializing the respective quote style. Selecting a quote style will make these hooks available to all quotation macros without expanding them. The expansion of `#3` will take place immediately before the opening outer quote is inserted, but inside the group formed by the quotation. `#4` will be expanded before the opening inner quote is inserted. It is advisable to avoid any global assignments in this context to prevent interference with other styles. Please note that if `#4` is used `#3` has to be given as well, even if the argument is empty.

Refer to table 3 for a list of all styles and variants defined by default. Note that these are the backend styles only, see also table 4 for a list of language aliases. If you need support for a language that is not supported by default, a new quote style is easily defined. A complete example will be given in section 6.6.

6.2 Defining quote aliases

Use `\DeclareQuoteAlias` to define a quote alias if the name of a quote style differs from the language name used by the babel package or if several languages share a common quote style:

```
\DeclareQuoteAlias[#1]{#2}{#3}
```

This macro takes one optional and two mandatory arguments:

`#1` quote style variant (optional)
`#2` quote style name
`#3` alias name

`\DeclareQuoteAlias` may be used in the configuration file or in the document preamble. If you define an alias for babel, the alias name must be identical to the language name used by the babel package, i.e. the expansion of `\languagename`.

Aliases		Quote style
Language alias	Intermediate alias	
american	–	english/american
USenglish	american	english/american
austrian	–	german/quotes
naustrian	austrian	german/quotes
british	–	english/british
UKenglish	british	english/british
canadian	–	english/american
danish	–	danish/quotes
english	–	english/american
french	–	french/quotes
german	–	german/quotes
ngerman	german	german/quotes
italian	–	italian/quotes
norwegian	–	norwegian/guillemets
norsk	norwegian	norwegian/guillemets
nynorsk	norwegian	norwegian/guillemets
swedish	–	swedish/quotes
swiss	–	german/swiss

Table 4: Language aliases defined by default

The style may be a backend style or another alias. As list of all aliases defined by default is given in table 4. Most language aliases refer directly to a backend style, but some point to an intermediate alias instead. The relation of aliases to backend styles is entirely technical, hence a language alias might very well point to a backend style that is completely unrelated on linguistic grounds.

6.3 Defining package options

Use `\DeclareQuoteOption` to create a new package option based on a key/value syntax. This macro takes one mandatory argument, the quote style name:

```
\DeclareQuoteOption{#1}
```

When using the new package option, the quote style will serve as the key. The value of this key can by any style variant defined for the respective quote style. This macro is available in the configuration file only.

6.4 Defining quote characters for PDF strings

Use `\DeclarePlainStyle` to change the characters used as quotation marks in PDF strings:

```
\DeclarePlainStyle{#1}{#2}
```

This macro takes two mandatory arguments:

`#1` outer quotation mark (opening and closing)
`#2` inner quotation mark (opening and closing)

`\DeclarePlainStyle` may be used in the configuration file or in the document preamble. By default, outer quotation marks will be replaced with straight double quotes and inner quotation marks with straight single quotes.

6.5 Configuring block quotations

You may adapt the default values used by the block quote macros to your requirements. Use `\setblockthreshold` to change the number of lines csquotes will use as a threshold when deciding whether a quotation should be typeset inline or as a block quote. This macro takes one argument which may be zero or a positive integer. Use `\setblockenvironment` to change the environment used for block quotes. This macro takes the name of the block quote environment as an argument. Both macros may be used in the configuration file, in the document preamble, or in the document body. By default, the threshold is set to five lines and the environment used for quotations exceeding this threshold is the `quote` environment provided by most Latex classes. If the `quote` environment is not defined, csquotes will provide a default definition. Here are two examples:

```
\setblockthreshold{10}
\setblockenvironment{quotation}
```

There are some additional hooks which may be used to customize the behavior of the block quote macros at a lower level. For both short and long quotations, the block quote macros will pass the text to be inserted after the quotation to an internal macro called `\blockcite`. This macro will do nothing by default, but it may be used as a hook to format the text after the quotation. When redefining `\blockcite`, keep in mind that it must take exactly one mandatory argument, the text to be inserted after the quotation. For example, you could automatically enclose the text after the quotation in parentheses or put it in a footnote by redefining `\blockcite` as follows:

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

If the quotation is long, the quoted text will also be wrapped in an additional environment called `quoteblock`. Like `\blockcite`, the `quoteblock` environment does nothing by default but it may be redefined if additional hooks are required to format the block quotation as desired. If the quotation is long, `\blockquote` will essentially expand as follows:

```
\begin{block quote environment}%
  \begin{quoteblock}%
    quoted text%
  \end{quoteblock}%
  \blockcite{text after quotation}%
\end{block quote environment}
```

The block quote macros for quotations in a foreign language insert the text after the quotation inside the block quote environment but outside the scope of

the language change. The quoteblock environment is inserted at the innermost level. Hence \foreignblockquote will essentially expand as follows:

```
\begin{block quote environment}%
  \begin{otherlanguage*}{language}%
    \begin{quoteblock}%
      quoted text%
    \end{quoteblock}%
  \end{otherlanguage*}%
  \blockcite{text after quotation}%
\end{block quote environment}
```

\hyphenblockquote will expand in a similar way, using the hyphenrules environment instead of otherlanguage*.

6.6 Adding a new quote style

This section provides a comprehensive example of how to define your own quote styles. The configuration file csquotes.cfg includes further examples. In order to add a quote style for an imaginary language called Newspeak, you might put the following in your configuration file:

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

Suppose there were two quote styles commonly used in Newspeak, an official one and an inofficial one often seen in underground publications. In this case, you would define the following two styles instead of the one above:

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

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

Since the official style is, well, official, it should be the default whenever we select the style newspeak:

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

Now suppose babel offered support for Newspeak, but this language was know to the babel package as otherspeak:

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

This is an example of an alias pointing to another alias defined before. If babel support was enabled and the document language was set to otherspeak, the above aliases would be expanded as follows:

```
otherspeak → newspeak → newspeak/official
```

The reason why we are using chained aliases instead of defining them in parallel will become clear in a moment. You may want to create a package option to select which of the two styles to use in a given document:

```
\DeclareQuoteOption{newspeak}
```

This will define a new package option called `newspeak`. The value of the option key may be any variant of the `newspeak` style defined in the configuration file. To select a variant, use the package option as follows. To select the default or the alternative style for the entire document:

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

To select the default or the alternative style with babel support:

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

The `newspeak` option will select a variant by redefining the `newspeak` alias. Since the `otherspeak` alias points to `newspeak` and not directly to any backend style, this setting will also have the desired effect if babel support is enabled.

7 Revision history

2.4 2004-11-01

Prevent use of `\RestoreQuotes` in preamble (4.3)
Fixed bug causing premature expansion of quote macros
Fixed bug suppressing kerning before closing quotes

2.3 2004-09-18

Reduced default kerning between adjoining curved quote marks
Fixed bug with `\DeclareQuoteStyle` in preamble

2.2 2004-07-13

Extended `\DeclareQuoteStyle` (6.1)
Added initialization hook for inner quotes (6.1)
Added support for middle inner quote marks (6.1)
Rearranged French quote styles, removing two variants (2.4; tab. 1, 3)
Added new style variant for French (2.4; tab. 1, 3)
Fixed bug causing stacking of reset hook for footnotes
Fixed bug preventing hyphenation in certain places
Fixed kerning issue specific to EC fonts

2.1 2004-06-15

Added internal environment `quoteblock` (6.5)
Disable active quotes in `\verb` and `verbatim` (5.2)
Disable active quotes in `\index` and `\glossary` (5.2)
Added support for language reset in footnotes (5.5)
Added package option and style variants for Norwegian (tab. 1, 3)
Rearranged quote styles and aliases (tab. 3, 4)
Removed some uncertain quote styles and aliases

2.0 2004-06-04

Added `\blockquote` (3.3)
Added `\foreignblockquote` (3.4)
Added `\hyphenblockquote` (3.4)
Added `\setblockthreshold` (6.5)
Added `\setblockenvironment` (6.5)
Added internal hook `\blockcite` (6.5)
Extended `\DeclareQuoteStyle` (6.1)
Added initialization hook for outer quotes (6.1)
Added support for middle outer quote marks (6.1)
Added support for kerning between adjoining quote marks (6.1)
Added package option and new style variants for French (2.4; tab. 1, 3)
Added package option and new style variant for Italian (tab. 1, 3)
Added new style variant for English (2.4; tab. 1, 3)
Added basic support for Omega (2.4)
Disable active quotes in math and display math mode (5.2)
Revised and improved error recovery (2.2, 5.1, 5.2)
Added `strict` option (2.2; tab. 1)

1.7 2004-05-14

Added `\setquotestyle` (3.5)
Modified `\DeclarePlainStyle` (6.4)
Modified quote handling in PDF strings (5.3)
Modified default French quote style

1.5 2004-02-27

Added support for nesting level reset in footnotes (5.5)

1.4 2003-12-13

Improved hyperref interface (5.3)

1.3 2003-12-11

Added `\MakeForeignQuote` (4.2)
Added `\MakeHyphenQuote` (4.2)
Added `\RestoreQuotes` (4.3)

1.0 2003-09-14

Initial public release