

## I. Specify packages, classes, macros or environments

Here's what you can type semantically.

<code>\tdoccls{myclass}</code> is for...	<code>\\</code>	myclass is for...
<code>\tdocpack{mypackage}</code> is for...	<code>\\</code>	mypackage is for...
<code>\tdocmacro{onemacro}</code> is for...	<code>\\</code>	\onemacro is for...
<code>\tdocenv{env}</code> produces...	<code>\\</code>	\begin{env} ... \end{env} produces...
<code>\tdocenv[<i>{[opt1]&lt;opt2&gt;}</i>]{env}</code>	<code>\\</code>	\begin{env}[opt1]<opt2> ... \end{env}
Just <code>\tdocenv*{env}</code> ...	<code>\\</code>	Just env...
Finally <code>\tdocenv*[<i>{[opt1]&lt;opt2&gt;}</i>]{env}</code> ...		Finally env...

**Remark I.1.** Unlike `\tdocinlatex`, `\tdocenv` and `\tdocenv*` macros don't color the text they produce. In addition, `\tdocenv{monenv}` produces `\begin{monenv} ... \end{monenv}` with spaces to allow line breaks if required.

### ⚠ Warning.

The optional argument of the `\tdocenv` macro is copied and pasted <sup>a</sup> when rendering. This may sometimes require the use of protective braces, as in the example above.

<sup>a</sup>Remember that almost anything is possible from now on.

## II. Origin of a prefix or suffix

To explain the names chosen, there is nothing like indicating and explaining the short prefixes and suffixes used. This is easily done as follows.

<code>\tdocpre{sup}</code> relates to...	<code>\\</code>	sup relates to...
<code>\tdocprewhy{sup.erbe}</code> means...	<code>\\</code>	sup·erbe means...
<code>\emph{\tdocprewhy{sup.er}}</code> for...		sup·er for...

**Remark II.1.** The choice of a full stop to split a word allows words with a hyphen to be used, as in `\tdocprewhy{bric.k-breaker}` which gives `bric.k-breaker`.