

I. Specify packages, classes, macros or environments

Here's what you can type semantically.

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

Remark. 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 ^a when rendering. This may sometimes require the use of protective braces, as in the example above.

^aRemember 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} relates to...</code>	<code>\\</code>	sup relates to...
<code>\tdocprewhy{sup.erbe} means...</code>	<code>\\</code>	sup•erbe means...
<code>\emph{\tdocprewhy{sup.er} for...}</code>		sup•er for...

Remark. 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`.