The FramedSyntax package

Claudio Beccari

`claudio.beccari(at)gmail.com`

Version v.0.2.8 – Last revised 2024-09-04.

Contents

Preface

The FramedSyntax package contains the necessary code to implement a framed block of text that is emphasised compared to the normal text. I found this technique very handy for writing the documented TEX files that describe the new packages that I create, and the user manuals of those packages. As a member of the Italian Tug I also wrote a number of thematic guides for the Italian users; they seem to appreciate that the syntax of LATEX, classes and packages are written with these framed emphasised blocks, clearly emerging form the surrounding text.

This is why I would like to share my package with the other TEX users. May be they would like to use the same approach. I perfectly know that there are other packages and classes to write down documented TEX files, such as those that traditionally have a `.dtx` file extension. Some of them are really powerful; I wonder why I keep using the docstrip.tex file and the ltxdoc class to write down my documented TEX files. The first simplest answer is “Because I am used to it”. The second is “Because I don’t like to use too powerful packages, such as, for example, tikz”; this package is a beautiful one, extremely powerful, but it is memory consuming, and computer blocking due to exhausted TEX working memory occurred to me more often than I could stand.

Dear User: if you do not like the name FramedSyntax downloaded with a full and updated TeX System installation, copy this file to your local `texmf` directory, and change the file name to another name that fits your preferences. Never change the names of the TEX System files even if you have the administrator privileges. In any case, since this work was inspired by a style file of mine written in Italian, this package already contains an alias ***Sintassi*** environment with the Italian name.

1 What the FramedSyntax.sty package can do

The user can use this package and can specify the package options; these include the colours of the frame, the text, the background; the frame thickness and colour; the frame-text gap; the text font family, series, shape, and size. These settings can be global or local, therefore you can select some of these settings for specific syntax instances.

Most often than not, you'll be using the default text fonts for the whole document. Please do not use the old Computer Modern fonts; they are not scalable, but may be chosen at a discrete set of sizes. The Latin Modern ones are step-wise continuously scalable, thanks to the scalable optical sizes they are built with. Other fonts, such as the Times, the Palatino, the Libertinus ones, and many others are continuously scalable over wide ranges.

While describing and commenting the code, you will read more detailed comments on such font issues.

In general the syntax frame fills up the whole `\textwidth`; in reality it should fill up the current `\linewidth`; within a list this frame will be shorter and indented as the list text.

This framed syntax box will not split across a page break; matter of facts, the next section example height is too large to fit in the page, and a large white vertical space filled this page, in order to have the framed box completely in within the next section. If you need long framed texts, you should preferably use the tcolorbox package (which relies on the extremely powerful tikz one) and performs much better than my simple package, in particular it can split its boxes across page breaks.

Warning: This small package does similar things as the wrapfig2 package, but such packages are not supposed to interfere with each other.

```
% !TEX TS-program = ⟨LaTeX based compiler⟩
% !TEX encoding = UTF-8 Unicode

\documentclass[⟨options⟩]{⟨class⟩}
... ⟨Packages⟩
\usepackage[⟨key=value options⟩]{FramedSyntax}
... ⟨Other packages⟩

\begin{document}
... ⟨A part of document body⟩
\begin{FramedSyntax}
... ⟨Some syntax lines⟩
\end{FramedSyntax}
... ⟨Further document body⟩

\end{document}
```

2 Usage example

The above framed syntax box contains a sample for using this package.[1]

In order to use this package a user should use this syntax. The above elementary example was done with the following source code.

```
\begin{FramedSyntax}
\texttt{\% !TEX TS-program = } \meta{LaTeX based compiler}
\texttt{\% !TEX encoding = UTF-8 Unicode}
~
\cs{documentclass}\oarg{options}\marg{class}
\dots\ \meta{Packages}
\cs{usepackage}\oarg{key=value options}\Marg{FramedSyntax}
\dots\ \meta{Other packages}
~
\cs{begin}\Marg{document}
\dots\ \meta{A part of document body}
\cs{begin}\Marg{FramedSyntax}
\dots\ \meta{Some syntax lines}
\cs{end}\Marg{FramedSyntax}
\dots\ \meta{Further document body}
~
\cs{end}\Marg{document}
\end{FramedSyntax}
```

3 Indexing

The syntax macros can send their arguments to an `.idx` file. If package imakeidx is loaded in the preamble and the `\makeindex` macro (with its own options) is used to activate the indexing process, the index is created without the need of manually running the necessary programs to transform the information collected into the `.idx` to the proper code into the `.ind` file.

For example the following source code

```
% !TEX TS-program = pdflatex
% !TEX encoding = UTF-8 Unicode
\documentclass[12pt]{article}
\usepackage{imakeidx}\makeindex
\usepackage{FramedSyntax}
\usepackage{kantlipsum}

\begin{document}
\kant[1]
\begin{FramedSyntax}
\cs{Large}
\end{FramedSyntax}
```

---

[1] You may notice that the framed box width is a little shorter than the `\linewidth`, exactly 10 pt shorter. This happens with this particular documentation class ltxdoc: I was not able to find the cause, but several tests with other classes proved to perform as expected.

```
\kant[2-3]
\begin{FramedSyntax}
\cs{small}
\end{FramedSyntax}

\kant[4]

\printindex

\end{document}
```

produces a three page article containing dummy test, with a framed syntax box in each page. Both framed commands, introduced by means of the `\cs` command, are listed in the third page, correctly pointung to page 1 end page 2. Users are suggested to copy the above short code, save it in a testing `.tex` file, and compile it with `pdfLaTeX`; with just one run the file gets compiled together with its index.

4 The documented code

Here the package code is defined and commented.

4.1 The necessary packages

FramedSyntax relies on a number of other packages; they are the following:

```
\IfPackageLoadedF{pict2e}{\RequirePackage{pict2e}}
\IfPackageLoadedF{etoolbox}{\RequirePackage{etoolbox}}
\IfPackageLoadedF{xkeyval}{\RequirePackage{xkeyval}}
\IfPackageLoadedF{xcolor}{\RequirePackage{xcolor}}
%
```

The last three packages are loadeed for obvious reasons, since they are connected to LaTeX programming and the package specific functionalities. May be the most puzzling one is pict2e; it is loaded in order to use its low level commands used to draw the frame with rounded corners, i.e. the lines and the corner arcs; they can be coloured and their thickness may be specified; they form closed paths that can be stroked in a specific colour or filled with another specific colour.

4.2 The syntax macros

These numerous commands are used to typeset in a specific style the command arguments, the files, classes, packages, options names; commands, and so on. If in any source file indexing is active, each command sends the relevant information to the `.idx` file.

All package internal commands have names that are prefixed with the string `FS`, so as to avoid conflicts with similar named commands defined by other classes or packages. These are not user commands, they are only for internal use. At the beginning of the ***FramedSyntax*** environment all user commands (that users input without this prefix) will be let to the internal commands, so that no interference exists and no clash errors take place.

Notice the non prefixed names: `\marg` and `\Marg` (and its siblings): with the lower case initial letter the syntax *mandatory* argument is printed in italics surrounded by angle brackets; with an uppercase initial letter the syntax *mandatory* argument is printed in teletype and is not surrounded by angle brackets. Similarly syntax *optional* arguments are typeset within square brackets but the initial letter case specifies the difference between an argument to be entered by the user, and an argument with a specific value.

Since initially I wrote the package for myself, some unusual object names are spelled in mixed languages: for example it is evident that `marg` stands for *mandatory argument* without any language mixture; the Italian word `stile` evidently stands for the English word *style*; similarly `allineamento` stands for *alignement*; `numeristyle` is in mixed languages but obviously it stands for *numbers style*; possibly the only one, that may puzzle a non Italian user, is the word `chiave` that stands for *key*.

Entries for a possible index are already created in such a way as to use the correct language for the index in Italian or the index in English. Should another user prefer to create the index in a different language, such a user is free to redefine in his/her document the indexing macros defined in this package¿

```
\providecommand*\FSmeta[1]{\textnormal{$\langle$\textit{#1}$\rangle$}}
\providecommand*\FSmarg[1]%
  {\textnormal{\texttt{\char123}\FSmeta{#1}\texttt{\char125}}}
\providecommand*\FSoarg[1]{\textnormal{\texttt{[}\FSmeta{#1}\texttt{]}}}
\providecommand*\FSArg[1]{\textnormal{\texttt{\{#1\}}}}
\let\FSMarg\FSArg
\providecommand*\FSOarg[1]{\textnormal{\texttt{[#1]}}}
\def\GT@splitargs#1,#2!{\def\@tempA{#1}\def\@tempB{#2}}
\providecommand\FSgarg[1]{\textnormal{\GT@splitargs#1!\%
  \texttt{(}\FSmeta{\@tempA}\texttt{,}\FSmeta{\@tempB}\texttt{)}}}
\providecommand*\FScomando[1]{\textnormal{\texttt{\string#1}}}
\providecommand*{\FScs}[1]%
{\textnormal{\texttt{\char92#1}\index{#1@\texttt{\char92#1}|textsc}}}
\let\FScsindex\FScs
%
\providecommand*\SFSambiente[2]{%
\FScomando{\begin}\FSmarg{#1}\FSoarg{#2}\,\dots
  \FScomando{\end}\FSmarg{#1}}
\providecommand*\DFSambiente[3]{%
  \FScomando{\begin}\FSmarg{#1}\FSoarg{#2}\FSoarg{#3}\,\dots
  \FScomando{\end}\FSmarg{#1}}
\providecommand*\BFSambiente[1]{\FScomando{\begin}\FSmarg{#1}}
\providecommand*\EFSambiente[1]{\FScomando{\end}\FSmarg{#1}}
\IfPackageLoadedF{cfr-lm}{\let\texttm\texttt \let\texttv\texttt}
%
\DeclareRobustCommand*\FSambstyle[1]{%
  {\normalfont\textsf{\slshape#1}}}
\DeclareRobustCommand*\FSclassstyle[1]{%
  {\normalfont\texttv{\itshape#1}}}
\DeclareRobustCommand*\FSfilestyle[1]{%
  {\normalfont\texttm{\texttt{#1}}}}
\DeclareRobustCommand*\FSpackstyle[1]{%
  {\normalfont\texttm{\ifbool{PDFTeX}{\textit}{\itshape}{#1}}}}%
\DeclareRobustCommand*\FSprogstyle[1]{{\normalfont\textsf{#1}}}
```

```
\DeclareRobustCommand*\FSprog[1]{\FSprogstyle{#1}%
\iflanguage{english}{\index{program!#1@\FSprogstyle{#1}|textsc}}%
   {\index{programma!#1@\FSprogstyle{#1}|textsc}}}%
\DeclareRobustCommand*\FSpack[1]{\FSpackstyle{#1}%
\iflanguage{english}{\index{package!#1@\FSpackstyle{#1}|textsc}}%
   {\index{pacchetto!#1@\FSpackstyle{#1}|textsc}}}
\DeclareRobustCommand*\FSclass[1]{\FSclassstyle{#1}%
 \iflanguage{english}{\index{class!#1@\FSclassstyle{#1}|textsc}}%
   {\index{classe!#1@\FSclassstyle{#1}|textsc}}}%
\DeclareRobustCommand*\file[1]{\FSfilestyle{#1}%
   \index{file!#1@\FSfilestyle{#1}|textsc}}%
\DeclareRobustCommand*\FSamb[1]{\FSambstyle{#1}%
   \iflanguage{english}%
   {\index{environment!#1@\FSambstyle{#1}|textsc}}%
   {\index{ambiente!#1@\FSambstyle{#1}|textsc}}}%
\DeclareRobustCommand*\FSopzstyle[1]{%
  {\normalfont\textsf{\slshape{#1}}}}%
\DeclareRobustCommand*\FScontastyle[1]{{\normalfont\texttm{#1}}}
\DeclareRobustCommand*\FSstilestyle[1]{{\normalfont\texttm{#1}}}
\DeclareRobustCommand*\FSnumeristyle[1]{{\normalfont\texttm{#1}}}
\DeclareRobustCommand*\FSumisurastyle[1]{{\normalfont\texttm{#1}}}
\DeclareRobustCommand*\FSchiavestyle[1]{{\normalfont\texttm{#1}}}
\DeclareRobustCommand*\FSdescrittorestyle[1]{{\normalfont\texttm{#1}}}
\DeclareRobustCommand*\FSposizionestyle[1]{{\normalfont\texttm{#1}}}
\DeclareRobustCommand*\FSallineamentostyle[1]{{\normalfont\texttm{#1}}}
%
\DeclareRobustCommand*\FSopz[1]{\opzstyle{#1}%
   \iflanguage{english}{\index{option!#1@\FSopzstyle{#1}|textsc}}%
   {\index{opzione!#1@\opzstyle{#1}|textsc}}}%
\DeclareRobustCommand*\FSconta[1]{\Fcontastyle{#1}%
  \iflanguage{english}{\index{counter!#1@\FScontastyle{#1}|textsc}}%
   {\index{contatore!#1@\FScontastyle{#1}|textsc}}}
\DeclareRobustCommand*\FSstile[1]{\FSstilestyle{#1}%
   \iflanguage{english}{\index{page style!#1@\FSstilestyle{#1}|textsc}}%
   {\index{stile della pagina!#1@\FSstilestyle{#1}|textsc}}}
\DeclareRobustCommand*\FSnumeri[1]{\FSnumeristyle{#1}%
   \iflanguage{english}{\index{numbering!#1@\FSnumeristyle{#1}|textsc}}%
   {\index{numerazione!#1@\FSnumeristyle{#1}|textsc}}}
\DeclareRobustCommand*\FSumisura[1]{\FSumisurastyle{#1}%
   \iflanguage{english}{measuring unit!#1@\FSumisurastyle{#1}|textsc}%
   {\index{unità di misura!#1@\FSumisurastyle{#1}|textsc}}}
\DeclareRobustCommand*\FSchiave[1]{\FSchiavestyle{#1}%
   \iflanguage{english}{\index{key!#1@\FSchiavestyle{#1}|textsc}}%
   {\index{chiave!#1@\chiavestyle{#1}|textsc}}}
\DeclareRobustCommand*\FSdescrittore[1]{\FSdescrittorestyle{#1}%
   \iflanguage{english}%
     {\index{column descriptor!#1@\FSdescrittorestyle{#1}|textsc}}%
     {\index{descrittore di colonna!#1@\FSdescrittorestyle{#1}|textsc}}}
\DeclareRobustCommand*\FSposizione[1]{\FSposizionestyle{#1}%
   \iflanguage{english}%
   {\index{floating object position!#1@\FSposizionestyle{#1}|textsc}}%
   {\index{posizione degli oggetti flottanti!#1@\FSposizionestyle{#1}|textsc}}}%
\DeclareRobustCommand*\FSallineamento[1]{FS\allineamentostyle{#1}%
   \iflanguage{english}%
   {\index{allignment code!#1@\FSallineamentostyle{#1}|textsc}}%
   {\index{codice di allineamento!#1@\FSallineamentostyle{#1}|textsc}}}
%
```

4.3 Register and colour definitions

Here we define some length and dimension registers. They will be used by the local commands and environments. Length `\insertwidth` might be already defined, therefore we test this possibility.

```
\newlength\SIfrthick
\newlength\SIfrgap
\newlength\SIfrwidth
\newlength\SIfrheight
\ifcsdef{insertwidth}{}{\newlength\insertwidth}
\newlength\SIXR \newlength\SIYD
\newlength\SIXL \newlength\SIYU
\newdimen\radius
\newdimen\framewidth

\definecolor{SIbackground}{rgb}{0.95,0.95,0.95}
\definecolor{SIframe}{rgb}{0.1,0.1,0.1}
\definecolor{SItext}{rgb}{0,0,0}

```

The default background color is a very light gray; the default frame colour is a very dark gray; the default text colour is just black. Such colours are deemed suitable for the application of the framed syntax box, but when necessary users can use the `\textcolor` command to emphasise specific phrases.

As it can be seen, these colours are defined by means of the colour defining macros of the xcolor package. They define some default colours; but the next commands redefine them at the user choice; actually the user shall specify some *key = value* options to the main syntax environment.

```
\def\SetSIbgd#1{\colorlet{SIbackground}{#1}}
\def\SetSIfrm#1{\colorlet{SIframe}{#1}}
\def\SetSItxt#1{\colorlet{SItext}{#1}}

```

4.4 Option settings

The next code is the strength of the package; the definition of the option settings. They form an option family defined by the angle brackets delimited string `sintassi`. Therefore they cannot be redefined by any other class or package unless the above family name is used.

```
\DeclareOptionX<sintassi>{fboxrule}[1pt]{\fboxrule=#1}
\DeclareOptionX<sintassi>{fboxsep}[1ex]{\fboxsep=#1}
\DeclareOptionX<sintassi>{framecolor}[SIframe]{\SetSIfrm{#1}}
\DeclareOptionX<sintassi>{backgroundcolor}[SIbackground]{\SetSIbgd{#1}}
\DeclareOptionX<sintassi>{fontstyle}[\normalfont]{#1}
\DeclareOptionX<sintassi>{radius}[\fboxsep]{\radius=#1}
\DeclareOptionX<sintassi>{insertionwidth}[\linewidth]{\insertwidth=#1}

\DeclareOptionX*{%
```

```
  \packageWarning{FramedSyntax}{‘\CurrentOption’ ignored}%
}

```

Notice that the option defining macro `\DeclareOptionX` is created by the keyvalue package in a simplified way, so as to imitate the LaTeX$_{2e}$ syntax.

4.5 Activate all options

With just two macros we activate all options of family `sintassi`

```
\ExecuteOptionsX<sintassi>{%
  fboxrule,
  fboxsep,
  framecolor,
  backgroundcolor,
  fontstyle,
  radius,
  insertionwidth
}
\ProcessOptionsX*\relax

```

4.6 A couple of useful local macros

The following macros are generally useful for the end users; the first one provides a simple means to scale the font size to any specified integer or fractional value. It comes useful when a line contained in a framed syntax box is too long and gets broken at the end of the box measure. According where the line is broken there are two ways to correct the source file:

- the user choses the best point where to break the line, inserts an explicit comment character (`\%`) at the end of the broken line and inserts a horizontal space at the beginning of the remaining part of the broken line.
- At the beginning of the environment body the user may insert a `\setfontsize` selecting the specific font size at a smaller integer or fractional value than the normal text size.

This `\setfontsize` macro accepts two different syntaxes in order to reach the same result: For example:

```
\setfontsize{9.5}[1.1]
\setfontsize[1.1]{9.5}
```

In any case the base line skip value is optional and its default value is 1.2 times the font size. Actually the optional value is not the base line skip value, but it is the `\linestretch` value, i.e. the ratio between the desired base line skip and the font size.

The second macro exploits the etoolbox functionalities to define a user macro that activates an internal native TeX macro protected by a `%` character.

```
\providecommand\setfontsize{}
\RenewDocumentCommand\setfontsize{O{1.2} m O{#1}}{%
  \fontsize{#2}{\fpeval{#1*#2}}\selectfont}

\providecommand\strippt{}
```

```
\renewcommand\strippt[1]{\csuse{strip@pt}#1}

```

4.7 The framed box requires several settings

The following macro `\framedbox` in a certain sense is the heart of the whole construction of the ***FramedSyntax*** environment. It requires 5 arguments, four mandatory ones and one optional. In order they are as follows

- `#1` mandatory: the width of the box/
- `#2` mandatory: the thickness of the frame.
- `#3` mandatory: the gap width between the frame and the contents of the box.
- `#4` optional: the radius of curvature of the box corners.
- `#5` mandatory: the text to be included within the framed box

Although the corner radius may be chosen, it is suggested to avoid specifying this optional parameter; its default value equals the gap width; if the frame thickness is zero, macro avoids tracing a frame with an invisible line.

```
\providecommand\framedbox{}
\RenewDocumentCommand\framedbox{m m m  O{#3} m}{%
\bgroup
  \dimen2=#1\relax
  \dimen0=\dimexpr#1-(#2+#3)*2\relax
  \setbox0\hbox{\parbox{\dimen0}{#5}}%
  \SIfrthick=#2\relax
  \SIXR=\dimexpr\wd0/2\relax \SIXL=-\SIXR\relax
  \SIYU=\dimexpr(\ht0+\dp0)/2\relax \SIYD=-\SIYU\relax
  \dimen4=#1\relax %                            box total width
  \dimen6=\dimexpr\ht0+\dp0+(#2+#3)*2\relax%   box total height
  \dimen8=#4\relax %                          corner arc radius
  \edef\SIbase{\strippt{\dimen4}}
  \edef\SIalt{\strippt{\dimen6}}
  \edef\SIxoff{\strippt{\dimexpr0.5\dimen4}}
  \edef\SIyoff{\strippt{\dimexpr0.5\dimen6}}
  \unitlength=1pt
  \begin{picture}(\SIbase,\SIalt)(-\SIxoff,-\SIyoff)
  \Frame*{\dimen4}{\dimen6}{\dimen8}%
  \ifdim\SIfrthick>0pt\Frame{\dimen4}{\dimen6}{\dimen8}\fi
  \put(0,0){\makebox(0,0){\box0}}%
  \end{picture}
\egroup
\ignorespaces}

```

4.8 The frame

The actual frame path may be filled with colour; the frame border may be coloured with its own specified colour; such colours are established by the FramedSyntax environment options or by the global options.

The `\Frame` command requires four parameters already computed by the `\framedbox` command; in any case they are the following:

`#1` optional: a boolean asterisk; if the asterisk is missing the frame contour is drawn with its colour `SIframe`; if the asterisk is present the contour is not drawn, but the framed box is filled with the `SIbackground` colour.

`#2` mandatory: overall box width.

`#3` mandatory: total box height.

`#4` mandatory: the corner arc radius.

The following code contains the low level pict2e commands to draw lines to form a closed path to be stroked or colour filled. They do not form a pict2e `\oval`, although the contour is similar, because the default `\oval` cannot be filled.

```
\providecommand\Frame{}
\RenewDocumentCommand\Frame{s m m m}{%
\bgroup
  \SIXR=\dimexpr#2/2\relax \SIXL=-\SIXR%
  \SIYU=\dimexpr#3/2\relax \SIYD=-\SIYU%
  \IfBooleanTF{#1}{\linethickness{0pt}\color{SIbackground}}%
  {\linethickness{\SIfrthick}\color{SIframe}}%
  \moveto(\SIXR,\SIYD+#4)%
  \circlearc{\SIXR-#4}{\SIYU-#4}{#4}{0}{90}%
  \circlearc{\SIXL+#4}{\SIYU-#4}{#4}{90}{180}%
  \circlearc{\SIXL+#4}{\SIYD+#4}{#4}{180}{270}%
  \circlearc{\SIXR-#4}{\SIYD+#4}{#4}{270}{360}%
  \closepath
  \IfBooleanTF{#1}{\fillpath}{\strokepath}%
\egroup\ignorespaces}

```

4.9 Filling the frame with the syntax text

We are almost at the end. Here we define another macro to put the syntax text inside the frame over the background colour. The next command `\includeframedtext` provides the necessary arguments to be passed to the lower level command `\framedbox`; the arguments are the following:

`#1` optional: box width; default value `\insertwidth`, which is the actual boxed text width; if it is larger than `\linewidth` it is reduced to this value; in this case users should revise they source code in order to fix this little problem.

`#2` mandatory: the syntax text to be framed.

`#3` optional: the local *key* = *value* options.

The code is the following.

```
\providecommand\includeframedtext{}
\RenewDocumentCommand\includeframedtext{O{\insertwidth} m O{}}%
{\bgroup
  \ExecuteOptionsX<sintassi>{#3}%
  \ifdimgreater{\insertwidth}{\linewidth}{\insertwidth=\linewidth}{}%
  \framedbox{\insertwidth}{\fboxrule}{\fboxsep}[\radius]{#2}%
\egroup}

```

4.10 The user environment

Originally this environmente was named ***Sintassi***, but in this package we name it ***FramedSintax***. The opening statement is very long, because it must contain all the `\let` commands that allow the user to use the command names without the `FS` prefix; such `\let` equivalences are local because the `FS` prefixes were introduced only to avoid clashes with possible definitions of the same names with other purposes.

But besides these numerous equivalences, this environment ***FramedSyntax*** determines the necessary parameters to pass to the service macros

The environment requires just two optional arguments.

`#1` optional: the font size. The default value is the current font size. This size should be specified only when the syntax lines are marginally longer than the available measure and they get broken close to their natural length. Please, remember that in order to have this size specification work properly, it is necessary to use continuously scalable fonts. This command will not have any useful effect with the original TEX default Computer Modern fonts; Latin Modern fonts, on the contrary, work very well.

`#2` optional: *key* = *value* options that locally override the global default ones or those that possibly were specified by the user when loading this package.

```
\ProvideDocumentEnvironment{FramedSyntax}{}{}{}
\RenewDocumentEnvironment{FramedSyntax}{o D(){}}
{%                                environment opening commands
\IfValueT{#1}{\setfontsize{#1}}%           font size setting
\ExecuteOptionsX<sintassi>{#2}%          key = value options
  \ifdimgreater{\insertwidth}{\linewidth}% width corrrection
    {\insertwidth=\linewidth}{}%
\framewidth=\insertwidth
%
\let\meta\FSmeta   \let\marg\FSmarg          \let\oarg\FSoarg
\let\Arg\FSArg     \let\Marg\FSMarg          \let\Oarg\FSOarg
\let\garg\FSgarg   \let\comando\FScomando    \let\cs\FScs
\let\csindex\FScs                  \let\Sambiente\SFSambiente
\let\Dambiente\DFSambiente
\let\Bambiente\BFSambiente   \let\Eambiente\EFSambiente
\let\amb\FSamb                \let\ambstyle\FSambstyle
\let\class\FSclass            \let\classstyle\FSclassstyle
\let\file\FSfile              \let\filestyle\FSfilestyle
\let\pack\FSpack              \let\packstyle\FSpackstyle
\let\prog\FSprog              \let\progstyle\FSprogstyle
\let\opz\FSopz                \let\opzstyle\FSopzstyle
\let\conta\FSconta            \let\contastyle\FScontastyle
\let\stile\FSstile            \let\stilestyle\FSstilestyle
\let\numeri\FSnumeri            \let\numerystyle\FSnumeristyle
\let\umisura\FSumisura          \let\umisurastyle\FSumisurastyle
\let\chiave\FSchiave            \let\chiavestyle\FSchiavestyle
\let\descrittore\FSdescrittore
            \let\descrittorestyle\FSDescrittorestyle
\let\posizione\FSposizione \let\posizionestyle\FSposizionestyle
\let\allineamento\FSallineamento
               \let\allineamentostyle\FSallineamentostyle

%
\begin{lrbox}{2}%                      open the necessary boxes
\begin{minipage}{%
  \dimexpr\framewidth-2\fboxrule-2\fboxsep}\obeylines
}{%                                environment closing commands
\end{minipage}\end{lrbox}%                close the above boxes
\begin{flushleft}%              set the syntax thext flush left
\includeframedtext{\box2}[#2]
\end{flushleft}}%

```

4.11 Backwards compatibility environment *Sintassi*

As said in the Preface, this package was inspired by a `.sty` file of mine written in Italian; I might conserve my older file but, since it is not so safe against macro definition clashes, in the several document source files I wrote with the previous implementation. I am going to change the name of the imported file, but I will conserve the multitude of instances of the old environment name, by an alias defined in this package.

```
\NewDocumentEnvironment{Sintassi}{o D(){}}%
  {\begin{FramedSyntax}[#1](#2)}{\end{FramedSyntax}}
```

.

Enjoy LaTeX