

domore

Some More Commands for Lists of Tokens

Uwe Lück*

November 19, 2012

Abstract

`domore.sty` is a package that enhances `dowith.sty`'s `\DoWith` (without assignments) and `\setdo` commands for applying something (e.g., `\do`) to each item of an “arglist”. Each item may consist of two or more arguments for a macro, and some “separator” material may be inserted between the applications to items. A practiced application has been generating inline lists of links that are separated by ‘ | ’. `domore.sty` is (to some extent) format-independent by means of the `plainpkg` and `stacklet` packages.

Related Packages: cf. `dowith.pdf`.

Keywords: Macro programming, programming structures, loops, list macros

Contents

| | | |
|-----|---|---|
| 1 | Making <code>domore.sty</code> Available | 2 |
| 2 | Remark on the Style of Code Documentation | 2 |
| 3 | Overview of Commands | 2 |
| 4 | Contents of <code>domore.sty</code> | 3 |
| 4.1 | Package File Header— <code>plainpkg</code> and Legalese | 3 |
| 4.2 | With L ^A T _E X, extend <code>dowith</code> 's <code>\setdo</code> | 3 |
| 4.3 | Auxiliaries | 4 |
| 4.4 | Enhanced <code>\DoWith</code> | 4 |
| 4.5 | Applications of <code>\DoWith</code> | 5 |
| 4.6 | Leaving and History | 6 |

*<http://contact-ednotes.sty.de.vu>

1 Making domore.sty Available

The domore package is a “plainpkg package” in the sense of the plainpkg¹ documentation that exhibits details of what is summarized here. Therefore:

- It is required that T_EX finds plainpkg.tex as well as stackrel.sty from the catcodes² bundle.
- In order to load domore.sty, type

`\usepackage{domore}` within a L^AT_EX document preamble,

`\RequirePackage{domore}` in a “plainpkg package”, or

`\input{domore.sty}`

... or perhaps `\input{<domore>.sty}`?

2 Remark on the Style of Code Documentation

In dowith.pdf, the documentation of the dowith package, in the section about “T_EX’s tokens,” I have tried to explain the difference between T_EX input code and the tokens that arise from it. In order to really understand what packages in the dowith bundle do, one should think of the behaviour of the *tokens*. For convenience however, I may rather fall back into the usual confusion here. After reading the documentation dowith.pdf of dowith.sty, you may be able to guess successfully what is meant below.

3 Overview of Commands

domore.sty provides a more powerful version of dowith.sty’s

`\DoWith{<repeat>}{<args>\StopDoing}`

acting on an “arglist” `<args>` where `<repeat>` may be more complex than with dowith.sty. Based on this, another variant `\DoWithMore` of `\DoWith` is provided where `<repeat>` may be a macro with more than one argument. With L^AT_EX e.g., `<repeat>` may be `\do` defined by

`\setdo[<digit>]{<opt>}{<replace>}`

an extension of dowith.sty’s `\setdo`. Further,

`\DoSeparateWith{<repeat>}{<sep>}{<args>\StopDoing}`

inserts “separator material” `<sep>` between the applications of `<repeat>` to the items in `<args>`. Another `\DoSeparateWithMore` combines the features of the

¹ctan.org/pkg/plainpkg

²ctan.org/pkg/catcodes

two previous macros. I have used this with `blog.sty` from the `morehype` bundle for generating inline lists of links, separated by something like ‘ | ’, in HTML documents.

As auxiliaries, variants `\@firstsecondoftwo` and `\@secondfirstoftwo` of L^AT_EX’s `\@firstofone` are introduced.

For details, see the comments to the package’s code below.

4 Contents of domore.sty

4.1 Package File Header—`plainpkg` and Legalese

```

1                                     \input plainpkg
2  \ProvidesPackage{domore}[2012/11/19 v0.3 dowith extended (UL)]
3  %% Copyright (C) 2012 Uwe Lueck,
4  %% http://www.contact-ednotes.sty.de.vu
5  %% -- author-maintained in the sense of LPPL below --
6  %%
7  %% This file can be redistributed and/or modified under
8  %% the terms of the LaTeX Project Public License; either
9  %% version 1.3c of the License, or any later version.
10 %% The latest version of this license is in
11 %%   http://www.latex-project.org/lppl.txt
12 %% We did our best to help you, but there is NO WARRANTY.
13 %%
14 %% Please report bugs, problems, and suggestions via
15 %%
16 %%   http://www.contact-ednotes.sty.de.vu
17 %%
```

4.2 With L^AT_EX, extend `dowith`’s `\setdo`

The original `dowith` offers `\setdo{⟨do⟩}` for defining a one-parameter macro `\do` expanding to `⟨do⟩`. The present package allows applying a `⟨digit⟩`-parameter macro (maybe `\do`, `⟨digit⟩` being 2, 3, or ...) to a list of “brace groups” where each brace group contains `⟨digit⟩` arguments. If L^AT_EX is present ...

```
18 \ifltx
```

... the following extension

```
\setdo[⟨digit⟩]{⟨do⟩}
```

of the basic `dowith` version can be used to define a `⟨digit⟩`-parameter macro `\do`. You also can equip `\do` with an initial optional argument by

```
\setdo[⟨digit⟩][⟨default⟩]{⟨do⟩}
```

The next two moves allow loading the package independently of `dowith` (overriding its definition of `\setdo`) as well as using the package with a format that has not defined `\do` before. The first parameter of `\do` may even be *optional*.

```

19 \let\setdo\relax \let\do\empty
20 \newcommand*{\setdo}[1][1]{\renewcommand\do[#1]}
21 \fi

```

4.3 Auxiliaries

`\@firstsecondoftwo{<balanced-1>}{<balanced-2>}` is a variant of L^AT_EX's `\@firstofone{<balanced>}` for *two* arguments. It just removes outer braces from each of the two arguments (provided it has outer braces), resulting in

`<balanced-1><balanced-2>`

```

22 \long\def\@firstsecondoftwo#1#2{#1#2}

```

`\@secondfirstoftwo{<balanced-1>}{<balanced-2>}` additionally interchanges the two arguments (after removing braces):

```

23 \long\def\@secondfirstoftwo#1#2{#2#1}

```

Our main application is using it as an extended `\expandafter` before `\fi`:

`\@secondfirstoftwo{<do>}\fi`

will expand to

`\fi<do>`

This won't work with `\else` in place of `\fi`.

4.4 Enhanced \DoWith

Here comes a more powerful variant of `dowith`'s `\DoWith`. Instead of iterating a single “command” `<cmd>` on an arglist `<args>` by

`\DoWith{<cmd>}{<args>}\StopDoing`

(cf. `dowith.pdf`), the present `\DoWith` can have a more complex first argument. If `<args>` consists of some brace groups the first of which is `<farg>` so that `<args>` is

`{<farg>}{<rgs>}`

—`<rgs>` being the remaining arglist—

`\DoWith{<repeat>}{<args>}\StopDoing`

works like

`<repeat>{<farg>}\DoWith{<repeat>}{<rgs>}\StopDoing`

and so on—a recursive explanation. Or if `<args>` is

`{<arg-1>}{<arg-2>}...{<arg-n>}`

(*n* items), the result is like

`<repeat>{<arg-1>}<repeat>{<arg-2>}...<repeat>{<arg-n>}`

The actual definition is:

```

24 \def\DoWith#1#2{%
25     \ifx\StopDoing#2\empty      %% not \@empty for Plain 2012/11/05
26     \else\@secondfirstoftwo{#1{#2}\DoWith{#1}}\fi}

```

In order to **use** the remaining definitions from **dowith together with the present package**, load `dowith.sty` before `domore.sty`.

4.5 Applications of \DoWith

\DoWith still is somewhat auxiliary. What I have used in practice, are the following definitions.

`\DoMore{<repeat>}{<args>\StopDoing}` with `<args>` as above “unpacks” each arglist item so that `<repeat>` may be a macro with more than one argument—say, `<digit>` arguments. Then `<f-arg>` or `<arg-1>`, as well as `<arg-2>` ... `<arg-n>`, should provide an arglist consisting of `<digit>` items.

```

27 \def\DoWithMore#1{\DoWith{\@firstsecondoftwo{#1}}}

```

Now I use metavariable `<do>` instead of `<repeat>`. We consider some “separator” material `<sep>` to be inserted between instances of applying `<do>` to an item of `<args>`. We want to get

$$\langle do \rangle \{ \langle arg-1 \rangle \} \langle sep \rangle \langle do \rangle \{ \langle arg-2 \rangle \} \langle sep \rangle \dots \langle sep \rangle \langle do \rangle \{ \langle arg-n \rangle \}$$

This is achieved simply by starting with

$$\langle do \rangle \{ \langle farg \rangle \}$$

and then proceeding as with

$$\backslash\text{DoWith}\{\langle sep \rangle \langle do \rangle \} \{ rgs \} \backslash\text{StopDoing}$$

And that’s what `\DoSeparateWith{<do>}{<sep>}{<args>\StopDoing}` does:

```

28 \def\DoSeparateWith#1#2#3{#1{#3}\DoWith{#2#1}}

```

`\DoSeparateWithMore{<do>}{<sep>}{<args>\StopDoing}` combines the two previous things, inserting separator material `<sep>` and unpacking the nested arglists:

```

29 \def\DoSeparateWithMore#1#2{%          %% wieder 2012/06/05
30     \DoSeparateWith{\@firstsecondoftwo{#1}}{#2}}

```

My main application is that `<do>` is a link macro with arguments `<target>` and `<text>` and that `<sep>` is ‘ | ’ (or some tie variant) to get a horizontal list of links like

$$\langle text-1 \rangle \mid \langle text-2 \rangle \mid \dots \mid \langle text-n \rangle$$

4.6 Leaving and History

```

31 \PopLetterCatAt
32 \endinput
33
34 VERSION HISTORY
35 v0.1    2012/01/17  developed in 'texblog.fdf'
36                (using \[re]newcommand*)
37 v0.2    2012/08/07  own file 'domore.sty', \def's only
38                2012/08/08  dealing with "more" \setdo
39 v0.3    2012/11/05  using 'plainpkg'; removing old % code
40                (see stored v0.2); auxiliaries \long
41                2012/11/06  doc.: more on \setdo (<digit>, opt. arg.),
42                usage with 'dowith' \strong
43                2012/11/18  doc.: adjusted for 'catchdq'; reworking for
44                \DoWith; \DoWithMore, \DoSeparateWith
45                2012/11/19  doc.: \DoSeparateWithMore
46

```