fancyitem

Sašo Živanović[*]

May 3, 2020

Abstract

This package allows the user to redefine the stock `\item`. It also provides an end-of-item hook.

1 Introduction

The idea of the package is that some specialized environments might be better off with a specialized `\item` command. For example, within an environment for linguistic examples, it might be handy to have an `\item` which makes the judgment mark stick out of the example, and pushes the (parenthesized) language identifier to the right. Like this:

```
\NewDocumentCommand\LinguisticItem{d()s}{%           % in the preamble
  \IfValueT{#1}{\AtEndItem{\hfill(#1)}}%
  \PlainItem
  \IfBooleanT{#2}{\mbox{}\llap{*}}%
  \ignorespaces
}
\FancyItem[1]{enumerate}{\LinguisticItem}
\begin{enumerate}[(1),labelsep=1em]                  % in the document
\item(English) This sentence is grammatical.
\item(English)* This sentence ungrammatical is.
\end{enumerate}
```

(1) This sentence is grammatical. (English)

(2) *This sentence ungrammatical is. (English)

Or perhaps, a listing environment which automatically enables math mode?

[*]e-mail: saso.zivanovic@guest.arnes.si; web: http://spj.ff.uni-lj.si/zivanovic/

```
\newcommand\MathItem{%                              % in the preamble
  \AtEndItem{$}%
  \PlainItem$%
}
\FancyItem[1]{itemize}{\MathItem}
\begin{itemize}                                     % in the document
\item (a+1)^2=a^2+2a+1
\item \exists x\colon Ax\wedge Bx
\end{itemize}
```

- $(a+1)^2 = a^2 + 2a + 1$
- $\exists x\colon Ax \wedge Bx$

Sky is the limit once you have control over your `\item`.

2 Usage

Load the package:

```
\usepackage{fancyitem}
```

Define an *item macro* (or several) in any way you like, naming it as you wish:

```
\def\myitem{\PlainItem\cmd{\myitem}: }                                  % TeX
\newcommand\myitemcommand{\PlainItem\cmd{\myitemcommand}: }             %
  LaTeX
\NewDocumentCommand\MyItemCommand{}{\PlainItem\cmd{\MyItemCommand}: }   %
  LaTeX3
```

Associate the item macro to an environment at a certain depth:

```
\FancyItem[1]{itemize}{\myitem}
\FancyItem[1]{enumerate}{\myitemcommand}
\FancyItem[2]{enumerate}{\MyItemCommand}
```

Make a list!

```
\begin{enumerate}
\item enumerated list at depth one
\item again
  \begin{itemize}
  \item first-level itemize
  \item again
  \end{itemize}
\item
  \begin{enumerate}
  \item enumerate at depth two
  \item again
  \end{enumerate}
\end{enumerate}
```

1. `\myitemcommand`: enumerated list at depth one
2. `\myitemcommand`: again
    - `\myitem`: first-level itemize
    - `\myitem`: again
3. `\myitemcommand`:
    (a) `\MyItemCommand`: enumerate at depth two
    (b) `\MyItemCommand`: again

As a special treat, the package provides a hook into the end of an item. The hook is set up by `\AtEndItem`. It makes most sense within an item macro, like this:

3 Implementation

Package identification and dependencies.

```
\ProvidesPackage{fancyitem}
\RequirePackage{etoolbox}
```

3.1 Auxiliary definitions

The user-defined *item macros* are accessed via control sequences which `\fancyitem@itemcs` expands to. The two parameters are the environment name (`#1`) and the environment depth (`#2`). The depth of 0 is a fallback for unregistered depths, see `\FancyItem` below.

```
\def\fancyitem@itemcs#1#2{fancyitem@item@#1@#2}
```

The following macro expands to the control sequence of the *end-of-item token list* associated to the current environment at the current depth.

```
\def\fancyitem@endcurritemcs{fancyitem@end@\@currenvir @\fancyitem@dp}
```

This package is normally meant to be used alongside package `enumitem`, so the depth of a list environment is retrieved from `enumitem`'s internal depth counters `enitdp@`⟨*environment name*⟩.

```
\def\fancyitem@dp{%
  \ifcsdef{enitdp@\@currenvir}{%
    \expandafter\the\csname enitdp@\@currenvir\endcsname
  }{}%
}
```

In the absence of `enumitem`, we provide its alternative names of standard LaTeX counters `\@itemdepth` and `\@enumdepth` ourselves.[1]

```
\@ifpackageloaded{enumitem}{}{%
  \let\enitdp@itemize\@itemdepth
  \let\enitdp@enumerate\@enumdepth
}
```

3.2 The core

This is our redefinition of standard `\item`. If the current environment has an associated item macro at the current depth, we use it: we first insert the end-of-item token list into the stream, and then execute the user-defined macro. Otherwise, we try to use the fallback item macro for the current environment (the "depth" 0). If this fails as well, we fall back to the standard LaTeX item (saved in `\PlainItem`).

```
\def\fancyitem@item{%
  \ifcsdef{\fancyitem@itemcs{\@currenvir}{\fancyitem@dp}}{%
    \fancyitem@doatenditem
    \@nameuse{\fancyitem@itemcs{\@currenvir}{\fancyitem@dp}}%
  }{%
    \ifcsdef{\fancyitem@itemcs{\@currenvir}{0}}{%
      \fancyitem@doatenditem
      \@nameuse{\fancyitem@itemcs{\@currenvir}{0}}%
    }{%
      \PlainItem
    }%
  }%
}
```

If the end-of-item token list for the current environment at the current depth exists, insert it into the stream and then clear it.

```
\def\fancyitem@doatenditem{%
  \ifcsdef{\fancyitem@endcurritemcs}{\fancyitem@doatenditem@}{}}
\def\fancyitem@doatenditem@{%
  \expandafter\the\csname\fancyitem@endcurritemcs\endcsname
  \csname\fancyitem@endcurritemcs\endcsname={}%
}
```

3.3 User interface

`\FancyItem` This macro registers a user-defined item macro (`#3`) for a given environment (`#2`) at a given depth (the optional argument `#1`). If the depth is 0 (or omitted), the given item macro will be used as a fallback item macro for this environment.

[1]If `enumitem` is loaded with option `loadonly`, the alternative names are not defined and we don't provide them either, assuming that the user knows what she is doing.

Note that the user-defined item macro should in principle call the standard LaTeX `\item`, now stored in `\PlainItem`. The item macro may call `\AtEndItem`; its argument will be executed at the end of the item.

```
\newcommand\FancyItem[3][0]{%
```

Associate the environment–depth dependent control sequence to the given item macro.

```
  \cslet{\fancyitem@itemcs{#2}{#1}}{#3}%
```

The end-of-item hook for the final item in a list requires special attention. While the modified `\item` command inserts the end-of-item token list for non-final items, this insertion must be performed by the end of environment for the final item. We use `etoolbox`'s `\AtEndEnvironment` to call `\fancyitem@doatenditem` at the end of every registered environment.

Given that every depth of an environment can have its own item macro, `\FancyItem` can be called several times for the same environment. But calling `\fancyitem@doatenditem` once at the end of an environment suffices, as this macro deals with depth internally. Before blindly appending it to the end-of-environment hook, we therefore check if it is already there. We do this using `\ifpatchable`. The test is not bullet-proof, but it should suffice. The test furthermore relies on the internals of `etoolbox` (that the end-of-environment hook is stored in `@end@#2@hook`), but the worse that can happen if something goes wrong is that `\fancyitem@doatenditem` will get executed several times at the end of an environment, which is harmless, as the end-of-item token list is cleared after usage.

```
\expandafter\ifpatchable\expandafter
  {\csname @end@#2@hook\endcsname}{\fancyitem@doatenditem}{}{%
    \AtEndEnvironment{#2}{\fancyitem@doatenditem}%
  }%
}
```

`\AtEndItem` This macro will usually be called within the item macro, to defer some code until the end of item. After ensuring that the end-of-item token list for the current environment at the current depth actually exists (if not, it is created), the macro stores the given code into the token list.

```
\def\AtEndItem#1{%
  \ifcsdef{\fancyitem@endcurritemcs}{}{%
    \expandafter\newtoks\csname\fancyitem@endcurritemcs\endcsname
  }%
  \csname\fancyitem@endcurritemcs\endcsname={#1}%
}
```

`\PlainItem` Redefine the standard LaTeX `\item`, saving the original into `\PlainItem`, so that it can be called by user-defined item macros.

```
\let\PlainItem\item
\let\item\fancyitem@item
```