

Package `highlightlatex` manual

Vincent Kuhlmann
vincent.kuhlmann@hotmail.com

March 18, 2021

Abstract

This package provides colored syntax highlighting for L^AT_EX source code, without aid from outside L^AT_EX. This is in response to the general trend that people often fall back to verbatim for displaying code. This package aims to make good looking L^AT_EX source code feasible for all users. For this, it builds further on the generic ‘listings’ package. A possible output is shown in Figure 1.

Contents

1	Getting started	2
1.1	Inline style	2
1.2	Block style	2
2	Macro <code>\updatehighlight</code>	2
2.1	Adding a command to a highlighting rule	2
2.2	Custom highlighting rules	3
2.2.1	Example	3
2.2.2	Specification	3
3	Global settings	4
4	Fragile breaking situations (like beamer frames)	4
5	Adding extra space	5
6	License	5
7	Credits	5

And look at this beautiful code

```
% Here is some code
\setcounter{secnumdepth}{1}
\begin{document}
  \section{My section (and Hi!)}

  \unknowncommand\knowncommand
  \inAccA\inAccB\inAccC
  \section  $\sqrt{2}$   $\leftarrow$  cool!

  Insert literal tildes like ~. Hi!
\end{document}
```

with some text after it.

Let’s also show square brackets:

```
% File: document.tex
\documentclass[a4paper]{article}
\usepackage{amsmath}

\begin{document}
  Insert math like  $\sqrt{2}$ .
  \inAccA\inAccB\inAccC
\end{document}
```

The first line was of the form `\documentclass[]{}.` Very interesting. Note the `$` are only green because we defined it as a keyword.

Go to <https://github.com/vkuhlmann/highlight-latex>

Figure 1: Output of ‘demo.tex’.

1 Getting started

After having added the package, you can add LaTeX in two ways.

1.1 Inline style

EXAMPLE Your file begins with a line of the form `\hll|\documentclass[]{}|`. The square brackets ...

The first non-space character following `\hll` delimits the argument to this command.

1.2 Block style

EXAMPLE Your basic document now looks like

```
\begin{highlightblock}[gobble=2]
  \documentclass[a4paper]{article}
  \begin{document}
    Hello world!
  \end{document}
\end{highlightblock}
```

To prevent indentation of our `highlightblock` (here one tab) to be shown as part of the code, the `gobble` parameter strips them off. Play around with it until everything looks right. I recommend to set this value globally using `\def\defaultgobble{2}`. You can still override it on a per-block basis, if necessary.

There are situations where width of the block could run out of the page. For example, when using beamer and storing a block as described in the section ‘Fragile breaking situations’, the normal full-width of a slide is assumed. If you use multiple columns, set the `linewidth` on the `highlightblock`. This can be a fraction of the total slide width available, `0.6\textwidth` is 60% of the width, or an absolute value, like `10em`, which seems to equal 20 characters.

There are more keys you can provide. Check the `listings` package documentation¹ for options available to the `lstlisting`-environment and `lstset` command.

2 Macro `\updatehighlight`

2.1 Adding a command to a highlighting rule

By default, only some LaTeX commands will be highlighted in blue. If there are others you need, like `\tableofcontents` and `\figref`, update the highlighting rules:

USE

```
\updatehighlight{
  name = default,
  add = {
    \tableofcontents, \figref
  }
}
```

The change will only affect code after it. I recommend issuing `updatehighlight` in your preamble (before the `\begin{document}`). In some situations you might want to change things mid-document. That’s possible too.

¹<https://www.ctan.org/pkg/listings>

2.2 Custom highlighting rules

2.2.1 Example

As shown in `demo.tex`, you can put any command or keyword you want to highlight in a different color. You do this with

EXAMPLE

```
\updatehighlight{
  % name: How you like to refer to it. Allows you to modify the style later.
  name = spotlight,
  color = orange,
  add = {
    \tableofcontents
  }
}
```

You can use the `xcolor` syntax for describing colors as well. If you find the orange too bright, you can replace it with `orange!90!black`: 90% orange, remaining is black. For more information on color definitions and name, refer to [LaTeX/Colors](https://en.wikibooks.org/wiki/LaTeX/Colors) on Wikibooks².

2.2.2 Specification

The argument to `\updatehighlight` is a key-value list. Keys are processed sequentially. For example, use `color` before `add` rather than after it, and a key can appear multiple times. Each one will be processed. You can merge any two `\updatehighlight` in one. No need to close and reopen `\updatehighlight` for each highlighting rule.

You might be tempted to add a blank line for clarity; that means a new paragraph to LaTeX, don't do it. Instead, just put a line with only a `%` sign. Spacing within the argument is often irrelevant. If you need a comma in the value, surround your value with braces.

name

Creates or modifies a named rule. This key is optional.

The default keys are `default`, which includes a bunch of basic commands, and has by default a dark blue color, and `structure`, which consists of `\begin` and `\end` and prints them in light blue.

classoffset

Sets the `listings` `classoffset` manually. Try to avoid this. Use **name** to refer to existing rules instead.

add

Adds a command (`\mycommand`) or keyword (`Hi!`) to the current rule. The value can contain multiple values by opening braces, and comma separating values within them.

remove

Removes a commands or keywords from the current rule. The value can contain multiple values by opening braces, and comma separating values within them.

²<https://en.wikibooks.org/wiki/LaTeX/Colors>

clear

Removes all commands and keywords from the current rule. Use without value, for example

EXAMPLE

```
\updatehighlight{
  name = default,
  clear
}
```

color

Specifies a color for the rule. Equivalent to specifying `style` instead, with value `\color{value}` where `value` is the value for the **color** key. So `color=red` and `style=\color{red}` are equivalent.

style

Specifies a style for the rule. A rule can have only one style. If you specify a style after `add` or `remove`, this starts a new (unnamed) rule. In practice, the only style which will probably work for you is just a color. For that, using the ‘color’ key is just a bit easier and neater. But hey, you have the option to set whatever style you want. :)

3 Global settings

There are some global parameters involved in the appearance:

OVERVIEW

```
\colorlet{curlyBrackets}{red!50!blue}
\colorlet{squareBrackets}{blue!50!white}
\colorlet{codeBackground}{gray!10!white}
\colorlet{comment}{green!40!black}
\def\defaultgobble{0}
```

Each line can be set independent of each other, and each shows its default value.

There are package options you can use as well:

frame (default: `lines`)

Specifies the frame you want around code. My favorites are `lines` and `none`. Check the listings package documentation³ for all possibilities.

noframe (use without value)

Equivalent to `frame=none`.

styleanywhere (use without value)

Overrides the default behavior that `style` starts a new style after commands like `add` and `remove`.

4 Fragile breaking situations (like beamer frames)

When passing command arguments around, or storing environment content, LaTeX interprets all characters. This includes seeing `\maketitle` in `\hll|\maketitle|` as a real command. To prevent this behavior, everything from `\verb`, to the `verbatim`-environment, to the `listings` package (which this package depends on) temporarily changes the interpretation of characters that are still to be read. The backslash before `\maketitle` in `\hll|\maketitle|` will be read as ‘just text’ (a *letter* technically).

³<https://www.ctan.org/pkg/listings>

When content has already been interpreted, like the `frame`-environment in `beamer` does, this trick can't be done anymore. Instead, you either need to *escape* code, or *pre-process* the code outside a fragile breaking situation.

Escaping is done by preceding the special character with a backslash. For example, `\hl| \documentclass[]{}|` becomes `\hl| \documentclass[]\{}|`.

For large code blocks, this is undesirable. Therefore, the package provides for a companion to the `highlightblock`-environment: surround it with a `saveblock` environment which takes a single argument: a name to assign it. We use it to refer to it later. For example:

EXAMPLE

```
\begin{saveblock}{basicfigure}
\begin{highlightblock}[linewidth=0.6\textwidth]
\begin{figure}
\includegraphics
[width=0.9\linewidth]
{myPlot.pdf}

\caption{My plot}
\label{fig:myplot}
\end{figure}
\end{highlightblock}
\end{saveblock}
```

Do this outside a fragile breaking situation. (For the `frame`-environment example, that means just before the `frame` for example.) Then, where you want to use it, use `\useblock{basicfigure}`. There is also a variant `\consumeblock{basicfigure}`. If you save many blocks, these will all remain loaded in memory till your PDF has fully generated. The `\consumeblock` works like `\useblock`, except the saved block is deleted from memory after its use. Note this can also result in unexpected behavior, for example animations in a `beamer` frame might need the code line to be executed multiple times. Use `\useblock` when you can't make the guarantee the last use of a block.

There is a separate demo for fragile breaking situations. You can find it at `deamerdemo/deamerdemo.tex`.

5 Adding extra space

By default, `highlight-latex` follows an approach where it minimizes spacing. This gives you full control over how tight or spacious your document looks. Just use commands like `\medskip` to add extra spacing. The package doesn't currently include an option to do it everywhere automatically.

6 License

The package is available under MIT License. See `LICENSE.txt`.

7 Credits

Thanks for minor fixes:

gemmaro

For any small fix, bug, feature request, unclarity etc., you're welcome to open an issue or pull-request on <https://github.com/vkuhlmann/highlight-latex/issues>

Thanks for thinking along!