The sclang-prettifier package*

Julien Cretel
`jubobs.tex at gmail.com`

2014/06/14

Abstract

Built on top of the listings package, the sclang-prettifier package allows you to effortlessly prettyprint SuperCollider source code in documents typeset with LaTeX & friends.

Contents

Introduction 3
1 Why this package? 3
2 sclang-prettifier in action 3

User's guide 3
3 Installation 4
3.1 Package dependencies 4
3.2 Installing sclang-prettifier 4
4 Getting started 4
4.1 Loading sclang-prettifier 4
4.2 Displayed listings 5
4.3 Standalone listings 5
4.4 Inline listings 5
5 Advanced customization 5
5.1 sclang-prettifier's key-value interface 5
5.2 Changing the font of your SuperCollider listings 6

Miscellaneous 6
6 Missing features and known issues 6
7 Bug reports and feature suggestions 6
8 Acknowledgments 7

Implementation 7

---

[*] This document corresponds to sclang-prettifier v0.1, dated 2014/06/14.

9 Preliminary checks 7
10 Package options 7
11 Required packages 8
12 Definition of the `SuperCollider` language 8
13 Symbols, classes and environment variables 9
14 Using into listings' hooks 9
15 Key-value interface 11
16 User-level font macro 11
17 `SuperCollider-IDE` style 12

Index **13**

Listing 1: Some dummy SuperCollider code

```
p.clear;

"Hello World!".postln;
~grains.addSpec(\tfreq, [1, 40, \exp]);
~grains.addSpec(\overlap, [0.1, 10, \exp]);
~grains.addSpec(\pos, [0, b.duration]);  // 3.43 is nice!
~grains.addSpec(\rate, [0.5, 2, 'exp']);
/*
  Dummy block comment.
*/
~grains = { |tfreq = 25, overlap = 6, pan = 0, amp = 0.2, pos =
    3.43,
   rate = 1|
   var trig = Impulse.ar(tfreq);
   TGrains.ar(2, trig, b, rate, pos, overlap / tfreq, pan, amp)
};
~grains.play;
```

Introduction

1 Why this package?

SuperCollider is a programming language for real-time audio synthesis and algorithmic composition. In February 2014, James Harkins, a SuperCollider user, enquired on TeX.SX (here and here) about the possibility of using the listings package to automatically highlight syntactic elements of the SuperCollider language such as symbols, environment variables, and classes, without having to list them manually as listings keywords. My answers to James's questions form the basis of this package.

2 sclang-prettifier in action

The sclang-prettifier package defines a listings style, called `SuperCollider-IDE`, that mimics the style of the SuperCollider IDE. For an example, see listing 1, which is adapted from this TeX.SX question.

The sclang-prettifier package automatically highlights the following syntactic elements of the SuperCollider language.

Keywords `var`

To-end-of-line and block comments `//3.43 is nice!`

Symbols `\tfreq`, `'exp'`, etc.

Environment variables `~grains`

Classes `Impulse`, `TGrains`, etc.

User's guide

3 Installation

3.1 Package dependencies

sclang-prettifier requires relatively up-to-date versions of packages textcomp, xcolor, and listings, all three of which ship with popular TEX distributions. It loads those three packages without any options.

3.2 Installing sclang-prettifier

Once the package gets officially released on CTAN, you should be able to install it directly through your package manager.

However, if you need to install sclang-prettifier manually, you should run

```
latex sclang-prettifier.ins
```

and copy the `sclang-prettifier.sty` file to a path where LATEX (or your preferred typesetting engine) can find it. To generate the documentation, run

```
pdflatex sclang-prettifier.dtx
makeindex -s gglo.ist -o sclang-prettifier.gls sclang-prettifier.glo
makeindex -s gind.ist -o sclang-prettifier.ind sclang-prettifier.idx
pdflatex sclang-prettifier.dtx
pdflatex sclang-prettifier.dtx
```

4 Getting started

As stated above, the sclang-prettifier package is built on top of the listings package. If you already are a seasoned listings user, you should feel right at home. If you're not, be aware that this user's guide makes use of some listings functionalities (such as key-value options) without describing their usage. For more details on those functionalities, you should consult the listings documentation.

4.1 Loading sclang-prettifier

Simply write

```
\usepackage{sclang-prettifier}
```

somewhere in your preamble.

You may want to load the listings and xcolor packages with some options; in that case, make sure those options are passed to those two packages *before* loading the sclang-prettifier package.

The sclang-prettifier package currently offers two options.

`framed`

Draws (by default) a dark gray frame around each listing that uses the `SuperCollider-IDE` style.

`numbered`

Prints (by default) line numbers in light gray to the left of each listing that uses the `SuperCollider-IDE` style.

4.2 Displayed listings

To typeset a SuperCollider listing embedded in your `tex` file, simply enclose it in an `lstlisting` environment, and load the `SuperCollider-IDE` style in the environment's optional argument, using listings' `style` key.

```
\begin{lstlisting}[style=SuperCollider-IDE]
  ...
\end{lstlisting}
```

4.3 Standalone listings

In practice, though, keeping your SuperCollider listings in external files—rather than embedding them in a `tex` file—is preferable, for maintainability reasons. To typeset a SuperCollider listing residing in an `sc` (or `scx`, or `sco`) file, simply invoke the `\lstinputlisting` macro, load the `SuperCollider-IDE` style in the environment's optional argument, and specify the path to the file in question in the mandatory argument.

```
\lstinputlisting[style=SuperCollider-IDE]{sample.sc}
```

4.4 Inline listings

You may want to typeset fragments of SuperCollider code within the main text of your document. For instance, you may want to typeset the `var` keyword in a sentence, in order to explain its usage. The `\lstinline` macro can be used for typesetting such inline code.

```
\lstinline[style=SuperCollider-IDE]|var|
```

Arguably, typing all this only to typeset such a simple SuperCollider keyword can rapidly become tedious. Fortunately, the listings allows you to define a character as a shorthand for inline code, via the `\lstMakeShortInline` macro. However, this character should ideally neither be used by the language itself nor occur elsewhere in your document. Unfortunately, because the SuperCollider language already uses most (all?) ASCII characters, your choice is limited... Proceed with caution. For more details about inline code, see subsection 4.17 in the listings manual.

5 Advanced customization

5.1 sclang-prettifier's key-value interface

The listings package provides a large number of options accessible via a nifty key-value interface, which is described in its excellent documentation. The sclang-prettifier package extends listings' key-value interface interface by defining three additional keys that allow you to customize the styles applied to SuperCollider symbols, environment variables, and classes, should you wish to do so. All three keys are prefixed by "`sc`", to help you distinguish them from native listings keys.

For each of the three keys described below, the value assigned to it in the `SuperCollider-IDE` style is indicated on the right-hand side.

`scsymbolstyle=`⟨*style*⟩ `\color[RGB]{0,113,0}`

This key determines the style applied to SuperCollider symbols. The last token can be a one-parameter command, such as `\textbf` or `\underbar`.

`scenvvarstyle=`⟨*style*⟩ `\color[RGB]{147,70,14}`

This key determines the style applied to SuperCollider environment variables. The last token can be a one-parameter command, such as `\textbf` or `\underbar`.

`scclassstyle=`⟨*style*⟩ `\color[RGB]{0,40,211}`

This key determines the style applied to SuperCollider classes. The last token can be a one-parameter command, such as `\textbf` or `\underbar`.

5.2 Changing the font of your SuperCollider listings

The sclang-prettifier package uses the Computer Modern typewriter font by default, which, arguably, is far from ideal. I encourage you to switch to your favourite "programmer font" instead.

For `pdflatex` users, sclang-prettifier conveniently provides a macro for easily selecting the Courier font—which is used by default by the SuperCollider IDE.

`\scttfamily`

selects the Courier font.

To use Courier in your SuperCollider listings, you must pass `\scttfamily` to listings' `basicstyle` key (*after* loading the `SuperCollider-IDE` style) and also—this is important—load the fontenc package with option `T1`:

```
\usepackage[T1]{fontenc}
```

Miscellaneous

6 Missing features and known issues

The sclang-prettifier currently does not highlight numbers as the SuperCollider IDE does. Highlighting numbers in listings in a robust manner is notoriously difficult; I might implement a solution in the future, if I ever find a good one.

7 Bug reports and feature suggestions

The development version of sclang-prettifier is currently hosted on GitHub at Jubobs/sclang-prettifier. If you find an issue in sclang-prettifier that this manual does not mention, if you would like to see a feature implemented in the package, or if you can think of ways in which the sclang-prettifier documentation could be improved, please add an entry to the repository's issue tracker on GitHub; alternatively, you can send me an email at jubobs.tex@gmail.com

8 Acknowledgments

Thanks to the developers of the listings package, without which sclang-prettifier would never have existed. I'm also in debt to many TeX.SX users for their help, encouragements, and suggestions. Thanks in particular to James Harkins, whose questions inspired me to write this package, and to Marco Daniel, Enrico Gregorio (egreg), and Heiko Oberdiek, whose contributions to TeX.SX proved particularly helpful for the development of this package.

Implementation

Be aware that, for "namespacing", the sclang-prettifier package uses, not a prefix, but the "`scpr`" suffix (preceded by an `@` character) throughout.

9 Preliminary checks

`\lstoptcheck@scpr` Because the listings options `noaspects`, `0.21`, and `savemem` are incompatible with sclang-prettifier, checking whether the listings package has been loaded with any of those options is a good idea; if so, we should issue an error. This macro checks whether listings was loaded with a given option and, if so, throws an error.

```
\newcommand\lstoptcheck@scpr[1]
{%
  \@ifpackagewith{listings}{#1}%
  {
    \PackageError{sclang-prettifier}%
      {incompatible listings' option #1}%
      {%
        Make sure the `listings' package
        doesn't get loaded with option `#1'%
      }
  }
  {}
}
```

We now use this macro to make sure that none of the problematic listings options has been passed to listings during an earlier loading of that package.

```
\lstoptcheck@scpr{noaspects}
\lstoptcheck@scpr{0.21}
\lstoptcheck@scpr{savemem}
```

10 Package options

Framed listings

`\ifframed@scpr@` This option draws (by default) a frame around each listing that uses the `SuperCollider-IDE` style.

```
\newif\ifframed@scpr@
\DeclareOption{framed}{\framed@scpr@true}
```

Numbered lines

`\ifnumbered@scpr@` This option prints (by default) line numbers to the left of each listing that uses the `SuperCollider-IDE` style.

```
\newif\ifnumbered@scpr@
\DeclareOption{numbered}{\numbered@scpr@true}
```

Draft This option is simply passed to listings.

```
\DeclareOption{draft}{\PassOptionsToPackage{\CurrentOption}{listings}}
```

Final This option is simply passed to listings.

```
\DeclareOption{final}{\PassOptionsToPackage{\CurrentOption}{listings}}
```

Discard undefined options We discard any other option passed to sclang-prettifier by the user and issue a warning.

```
\DeclareOption*%
{%
  \OptionNotUsed
  \PackageWarning{sclang-prettifier}{Unknown ‘\CurrentOption’ option}
}
```

Process options

```
\ProcessOptions\relax
```

11 Required packages

The sclang-prettifier package require three packages without any package option: the textcomp package, in order to use listings' `upquote` key; the xcolor package, in order to color our SuperCollider code; and, of course, the listings package.

```
\RequirePackage{textcomp}[2005/09/27]
\RequirePackage{xcolor}[2007/01/21]
\RequirePackage{listings}[2013/08/26]
```

12 Definition of the `SuperCollider` language

Language name

`\language@scpr` To avoid code duplication in this package file, we define a macro that expands to the name of our new language, `SuperCollider`.

```
\newcommand\language@scpr{SuperCollider}
```

`\languageNormedDefd@scpr` However, because listings "normalizes" language names internally, we also need to define a macro that expands to the normalized name of the new language.

```
\expandafter\lst@NormedDef\expandafter\languageNormedDefd@scpr%
  \expandafter{\language@scpr}
```

Language definition We can now define our new listings language, using some `\expandafter` trickery on `\lstdefinelanguage`.

```
\expandafter\expandafter\expandafter\lstdefinelanguage\expandafter
{\language@scpr}
{%
  morekeywords  = {var},
  alsoletter    = \\~,
  alsoother     = @,
  sensitive     = true,
  morecomment   = [l]{//},
  morecomment   = [s]{/*}{*/},
  morestring    = [s]{"}{"},
  moredelim     = [s][\symbolStyle@scpr]{'}{'},
}[keywords,strings,comments]
```

13 Symbols, classes and environment variables

Storing relevant characters To detect whether an identifier is an environment variable or a symbol, we will need to test whether the identifier in question starts with a tilde or a backslash, respectively. listings developer's guide tells us that the only safe way to test against a character is to store it in a macro using listings' internal macro `\lst@SaveOutputDef`.

`\tilde@scpr` We save the tilde character thus.

```
\lst@SaveOutputDef{`~}\tilde@scpr
```

`\dollar@scpr` We save the dollar-sign character thus.

```
\lst@SaveOutputDef{`$}\dollar@scpr
```

No need for such definition for the backslash: listings already stores the backslash in a macro called `\lstum@backslash`.

14 Using into listings' hooks

We apply some necessary patches in two listings' hooks; but first, we define a couple of helper macros.

Helper macros

`\getfirstchar@scpr`
`\getfirstchar@@scpr`
`\firstchar@scpr`
Of these three helper macros, the first two macros extract the first character token in a given sequence of character tokens and store it in the third macro. This approach is adapted from this TeX.SX answer by Marco Daniel.

```
\newcommand\getfirstchar@scpr{}
\newcommand\getfirstchar@@scpr{}
\newcommand\firstchar@scpr{}
\def\getfirstchar@scpr#1{\getfirstchar@@scpr#1\relax}
\def\getfirstchar@@scpr#1#2\relax{\def\firstchar@scpr{#1}}
```

`Output` (See the listings documentation for more details on this hook.)

`\addedToOutput@scpr` We add this macro (initially empty) to listings' `Output` hook.

```
\newcommand\addedToOutput@scpr{}
\lst@AddToHook{Output}{\addedToOutput@scpr}
```

`\currentchar@scpr` This count is used to test a character token against A-Z.

```
\newcount\currentchar@scpr
```

`\@ddedToOutput@scpr` The `\addedToOutput@scpr}` macro is let to this one under certain conditions (more details follow).

```
\newcommand\@ddedToOutput@scpr
{%
```

If we're in listings' processing mode…

```
  \ifnum\lst@mode=\lst@Pmode%
```

… we save the first character token in the identifier being processed to a macro called `\firstchar@scpr`.

```
    \expandafter\getfirstchar@scpr\expandafter{\the\lst@token}%
```

If that token is a backslash, we apply the style associated to symbols.

```
    \expandafter\ifx\firstchar@scpr\lstum@backslash%
      \let\lst@thestyle\symbolStyle@scpr%
```

If that token is a dollar sign, we have a SuperCollider "character". we apply the style associated to symbols (as in the SuperCollider IDE).

```
    \else
      \expandafter\ifx\firstchar@scpr\dollar@scpr%
        \let\lst@thestyle\symbolStyle@scpr%
```

If that that token is a tilde, we apply the style associated to environment variables.

```
      \else
        \expandafter\ifx\firstchar@scpr\tilde@scpr%
          \def\lst@thestyle{\envvarStyle@scpr}%
```

Otherwise, if that character is a capital letter (A-Z), we apply the style associated to classes.

```
        \else
          \currentchar@scpr=65
          \loop
            \expandafter\ifnum%
            \expandafter`\firstchar@scpr=\currentchar@scpr%
              \let\lst@thestyle\classStyle@scpr%
              \let\iterate\relax%
            \fi
            \advance\currentchar@scpr by \@ne%
            \unless\ifnum\currentchar@scpr>90%
          \repeat%
        \fi
      \fi
    \fi
  \fi
```

Whatever style was applied, we still check whether the identifier is a keyword; if it is one, the keyword style is applied to it.

```
  \lsthk@DetectKeywords%
}
```

`PreInit` (See the listings documentation for more details on this hook.) Because the `\lst@AddToHook` affects hooks globally (i.e. for all listings), we must apply our patches only when required, i.e. in listings that use `SuperCollider`, and not in others. The `PreInit`, which is called at the very beginning of each listing, is where we do that. We check whether `\lst@language` and `\languageNormedDefd@scpr` expand (once) to the same replacement text and only apply our patches under that condition.

```
\lst@AddToHook{PreInit}
{%
  \ifx\lst@language\languageNormedDefd@scpr%
    \let\addedToOutput@scpr\@ddedToOutput@scpr%
  \fi
}
```

15 Key-value interface

We extend listings' key-value interface by defining several additional keys, which we will use to define a style similar to that of the SuperCollider IDE, and which will allow the user to customize the style of their SuperCollider listings.

Symbol style

`scsymbolstyle` `\classStyle@scpr` This key determines the style applied to SuperCollider symbols.

```
\newcommand\symbolStyle@scpr{}
\lst@Key{scsymbolstyle}\relax%
  {\def\symbolStyle@scpr{#1}}
```

Environment-variable style

`scenvvarstyle` `\envvarStyle@scpr` This key determines the style applied to SuperCollider environment variables.

```
\newcommand\envvarStyle@scpr{}
\lst@Key{scenvvarstyle}\relax%
  {\def\envvarStyle@scpr{#1}}
```

Class style

`scclassstyle` `\classStyle@scpr` This key determines the style applied to SuperCollider classes.

```
\newcommand\classStyle@scpr{}
\lst@Key{scclassstyle}\relax%
  {\def\classStyle@scpr{#1}}
```

16 User-level font macro

`\scttfamily` This user-level macro can be used for selecting the Courier font family, which is used by default in the SuperCollider IDE (v3.6.6, at least), and which, contrary to TeX default font family (Computer Modern), comes with a boldface version.

```
\newcommand\scttfamily{\fontfamily{pcr}\selectfont}
```

17 `SuperCollider-IDE` style

The `SuperCollider-IDE` style mimics the default style of the SuperCollider IDE.

`\toks@scpr` We allocate a token list register in which we store settings that we'll use to define the style.

```
\newtoks\toks@scpr
\toks@scpr=%
{
  language                = \languageNormedDefd@scpr,
  basicstyle              = \color{black}\ttfamily\normalsize,
  breaklines              = true,
  showspaces              = false,
  showstringspaces        = false,
  upquote                 = true,
  rulecolor               = \color{black!67},
  numberstyle             = \color{black!33},
  keywordstyle            = \color[RGB]{000,045,231}\bfseries,
  commentstyle            = \color[RGB]{202,018,000}         ,
  stringstyle             = \color[RGB]{095,095,095}         ,
  scsymbolstyle           = \color[RGB]{000,113,000}         ,
  scenvvarstyle           = \color[RGB]{147,070,014}         ,
  scclassstyle            = \color[RGB]{000,040,211}         ,
}
```

```
\ifframed@scpr@
  \toks@scpr\expandafter{\the\toks@scpr frame=single,}
\fi
\ifnumbered@scpr@
  \toks@scpr=\expandafter{\the\toks@scpr numbers=left,}
\fi
\begingroup\edef\@tempa{\endgroup
  \noexpand\lstdefinestyle{SuperCollider-IDE}{\the\toks@scpr}
}\@tempa
```

Change History

v0.1
General: Initial release. 1

Index

Symbols
`\@ddedToOutput@scpr` . . . . <u>57</u>, 89

A
`\addedToOutput@scpr` . . . . <u>54</u>, 89

C
`\classStyle@scpr` . . . . 74, <u>92</u>, <u>98</u>
`\currentchar@scpr` . . <u>56</u>, 70, 73, 77, 78

D
`\dollar@scpr` . . . . <u>48</u>, 64

E
`\envvarStyle@scpr` . . . . 68, <u>95</u>

F
`\firstchar@scpr` . . . . <u>49</u>, 61, 64, 67, 73

G
`\getfirstchar@@scpr` . . . . <u>49</u>
`\getfirstchar@scpr` . . . . <u>49</u>, 60

I
`\ifframed@scpr@` . . . . <u>17</u>, 120
`\ifnumbered@scpr@` . . . . <u>19</u>, 123

K
keys
  `scclassstyle` 11
  `scenvvarstyle` 11
  `scsymbolstyle` 11

L
`\language@scpr` . . . . <u>32</u>, 34, 36
`\languageNormedDefd@scpr` . <u>33</u>, 88, 105
`\lstoptcheck@scpr` . . . . <u>1</u>, 14–16

S
`\scttfamily` . . . . <u>101</u>
`\symbolStyle@scpr` . . 45, 62, 65, 92, 94

T
`\tilde@scpr` . . . . <u>47</u>, 67
`\toks@scpr` . . . . <u>102</u>, 121, 124, 127

U
unknown
  `\scttfamily` 6
  `scclassstyle` 6
  `scenvvarstyle` 6
  `scsymbolstyle` 6