Omikhleia's packages for SILE

Table of Contents

Chapter 1
Packages

1.1 epigraph: a lightweight epigraph environment

The **epigraph** package for SILE can be used to typeset a relevant quotation or saying as an epigraph, usually at either the start or end of a section. Various handles are provided to tweak the appearance.[1]

The **epigraph** environment typesets an epigraph using the provided text. An optional source (author, book name, etc.) can also be defined, with the **\source** command in the text block. By default the epigraph is placed at the right hand side of the text block, and the source is typeset at the bottom right of the block.

> lorem ipsum dolor sit amet consetetur sadipscing elitr sed diam nonumy eirmod tempor invidunt ut labore et dolore.
>
> *The Lorem Ipsum Book.*

Without source:

> lorem ipsum dolor sit amet consetetur sadipscing elitr sed diam nonumy eirmod tempor invidunt ut labore et dolore.

The default width for the epigraph block is **epigraph.width**. A **width** option is also provided to override it on a single epigraph[2]. It may be set to a relative

1. This is very loosely inspired from the LaTeX package by the same name.
2. Basically, all global settings are also available as command options (or reciprocally!), with the same name but the namespace left out. For the sake of brevity, we will therefore omit the namespace from this point onward.

value, e.g. 80 percents the current line width:

> lorem ipsum dolor sit amet consetetur sadipscing elitr sed diam nonumy eirmod tempor invidunt ut labore et dolore.

Or, pretty obviously, with a fixed value, e.g. **8cm**.

> lorem ipsum dolor sit amet consetetur sadipscing elitr sed diam nonumy eirmod tempor invidunt ut labore et dolore.

The vertical skips are controlled by **beforeskipamount**, **afterskipamount**, **sourceskipamount**. The latter is only applied if there is a source specified and the epigraph doesn't show a rule (see further below).

In the following example, the two first options are set to **0.5cm** and the source skip is set to **0**.

> lorem ipsum dolor sit amet consetetur sadipscing elitr sed diam nonumy eirmod tempor invidunt ut labore et dolore.
>
> *The Lorem Ipsum Book.*

By default, paragraph indentation is inherited from the document. It can be tuned with **parindent**, e.g. **1em**.

> lorem ipsum dolor sit amet consetetur sadipscing elitr sed diam nonumy eirmod tempor invidunt ut labore et dolore.
>
> lorem ipsum dolor sit amet consetetur sadipscing elitr sed diam nonumy eirmod tempor invidunt ut labore et dolore.

A rule may be shown below the epigraph text (and above the source, if present — in that case, the vertical source skip amount does not apply). Its thickness is controlled with **rule** being set to a non-null value, e.g. **0.4pt**.

> lorem ipsum dolor sit amet consetetur sadipscing elitr sed diam nonumy eirmod tempor invidunt ut labore et dolore.
>
> ---
>
> *The Lorem Ipsum Book.*

Likewise, without source:

lorem ipsum dolor sit amet consetetur sadipscing elitr sed diam nonumy eirmod tempor invidunt ut labore et dolore.

By default, the epigraph text is justified. This may be changed setting **ragged** to **true**.

The text is then ragged on the opposite side to the epigraph block, so on the left for a right-aligned block.

lorem ipsum dolor sit amet consetetur sadipscing elitr sed diam nonumy eirmod tempor invidunt ut labore et dolore.

The Lorem Ipsum Book.

It would be ragged on the right for a left-aligned epigraph block.

lorem ipsum dolor sit amet consetetur sadipscing elitr sed diam nonumy eirmod tempor invidunt ut labore et dolore.

The Lorem Ipsum Book.

Here, we introduced the **align** option, set to **left**. All the settings previously mentioned also apply to left-aligned epigraphs, so we can of course tweak them at convenience.

lorem ipsum dolor sit amet consetetur sadipscing elitr sed diam nonumy eirmod tempor invidunt ut labore et dolore.

The Lorem Ipsum Book.

It is also possible to offset the epigraph from the side (left or right) it is attached to, with the **margin** option, e.g. **0.5cm**:

lorem ipsum dolor sit amet consetetur sadipscing elitr sed diam nonumy eirmod tempor invidunt ut labore et dolore.

The Lorem Ipsum Book.

If you want to specify what styling the epigraph environment should use, you can redefine the **epigraph:style** style. By default it will be the same as the surrounding document, just smaller. The epigraph source is typeset in italic by default. It can be modified too, by redefining **epigraph:source:style**.[3]

> *lorem ipsum dolor sit amet consetetur sadipscing elitr sed diam nonumy eirmod tempor invidunt ut labore et dolore.*
>
> The Lorem Ipsum Book.

As final notes, the epigraph source is intended to be short by nature, therefore no specific effort has been made to correctly handle sources longer than the epigraph block or even spanning on multiple lines.

Before anyone asks, the alignment options for an epigraph are to the left or to the right of the frame only; notably, there is no intention to support centering. This author does not think centered epigraphs are typographically sound.

For the curious-minded, it turns out that nested epigraphs do work somewhat as intended. Your mileage may vary depending on the combination of settings. This is not expected to be a highly requested feature and has not been thoroughly tested.

> lorem ipsum dolor sit amet consetetur sadipscing elitr sed diam nonumy eirmod tempor invidunt ut labore et dolore.
>
> > Words — so innocent and powerless as they are, as standing in a dictionary, how potent for good and evil they become in the hands of one who knows how to combine them.
> >
> > *Nathaniel Hawthorne*
>
> lorem ipsum dolor sit amet consetetur sadipscing elitr sed diam nonumy eirmod tempor invidunt ut labore et dolore.
>
> ---
>
> *The Lorem Ipsum Book.*

3. Refer to our **styles** package for details on how to set and configure style specifications.

1.2 textsubsuper: native superscripts & subscripts

The **textsubsuper** package provides two commands, **\text:superscript{**⟨*content*⟩**}** and **\text:subscript{**⟨*content*⟩**}**.

They will use the native superscript or subscript characters, respectively, in a font, when available, instead of scaling, raising or lowering characters, because most of the time the former will obviously look much better.

As of superscripts, that could for a number (e.g. in footnote calls), but also for letters (e.g. in century references in French, IV^{e}; or likewise in sequences in English, 12^{th}).

As of subscripts, the most familiar example is in chemical formulas, for example H_2O or $C_6H_{12}O_6$.

These commands do so by trying the +**sups** font feature for superscripts, and the +**subs** or +**sinf** feature for subscripts.

If the output is not different than *without* the feature, it implies that this OpenType feature is not supported by the font (such as the default Gentium Plus font, that does not have these font features[4]): Scaling and raising or lowering is then applied.

By nature, this package is *not* intended to work with multiple levels of super- or subscripts. Also note that it tries not to mix up characters supporting the features with those not supporting them, as it would be somewhat ugly in most cases, so the scaling methods will be applied if such a case occurs.

1.3 abbr: a few shorthands & abbreviations

This package defines a few shorthands and abbreviations that its author often uses in articles or book chapters.

The **\abbr:nbsp** command inserts a non-breakable space. It is stretchable and shrinkable as a normal inter-word space by default, unless setting the **fixed** option to true.

The **\abbr:no:fr** and **\abbr:no:en** commands prepend a correctly typeset issue number, for French and English respectively, that is n^{o} 5 and no. 5.

The **\abbr:nos:fr** and **\abbr:nos:en** commands are the same as the previous commands, but for the plural, as in n^{os} 5–6 and nos. 5–6.

The **\abbr:no** and **\abbr:nos** invoke the appropriate command depending on the current language, nos. 12, 13.

4. Though it does include, however, some of the Unicode super- and subscript characters, but this very package does not try to address such a case.

The **\abbr:vol** acts similarly for volume references, that is vol. 4, just ensuring the space in between is unbreakable.

The **\abbr:page** does the same for page references, as in p. 159, but also supports one of the following boolean options: **sq**, **sqq** and **suiv**, to indicate subsequent page(s) in the usual manner in English or French, as in p. 159 *sq.*, p. 159 *sqq.* or p. 159 et suiv. Note that in these cases, a period is automatically added.

The **\abbr:siecle** command formats a century according to the French typographical rules, as in Ier or IVe, XVe and XIXe.

1.4 couyards: printer's ornaments

Typographers of the past relied on a number of ornaments to make their books look nicer.

A *cul-de-lampe* (plural *culs-de-lampe*) is a typographic ornament, sometimes called a *pendant*, marking the end of a section of text. The term comes from French, for "bottom of the lamp" (from the usual shape of the ornament). In French typography, they are also called *couillards* or *couyards* (apparenttly, erm… after a body part, really). It may be a single illustration or assembled from fleurons.

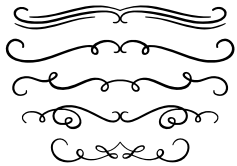
A *fleuron* is a typographic element or glyph, used either as a punctuation mark or as an ornament for typographic compositions. Fleurons, hence the name, which also derives from French, are usually stylized forms of flowers or leaves. The Unicode standard defines several such glyphs, as ❦ (U+2766, floral heart) and rotated versions of it, ❧ (U+2767) and ☙ (U+2619). Unicode version 7 even defines many more glyphs of that type.

In typography, these glyphs are also generically called *dingbats* or, when used as a section divider, a *dinkus*. One usual glyph used in the latter case is the *asterism,* ⁂ (U+2042).

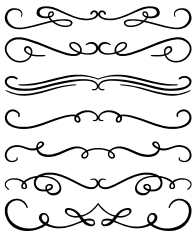
Whether fonts support these glyphs and how they render them is another topic. (For instance, here above, we had to switch to the Symbola font for the so-called floral hearts.)

This author, however, wanted a bit more variety with some old-fashioned ornaments independent from the selected font. The present **couyards** package therefore defines a few such ornaments, with the command **\couyard[type=⟨*n*⟩]**, where *n* is a number between 1 and 7.

Without any other option, the ornaments have a fixed height which can be overridden with the **height=**⟨*length*⟩ option, the default being 0.9em (also corresponding to **height=default**).



Alternately, one can set a width with the **width=**⟨*length*⟩ option, either to some length value or to **default** (7em).



These two options are exclusive, so as to keep a proper aspect ratio.

1.5 styles: (experimental) style specifications

The **styles** package can be used to easily define styling specifications. It is intended to be used by other packages or classes, rather than directly.

If one looks at the default/standard packages or classes in SILE, something may seem wrong (though of course it is a matter of taste and could be debated): (1) Many commands have "hooks", such as **pullquote:font**, **book:chapterfont**, or whatever. But what if ones also wants, for instance, to specify a color? Of course, in many cases, the hook could be redefined to apply that wanted color to the content… But, er, isn't it called **xxx:font**? Something looks amiss. (2) Those hooks often have fixed definitions, e.g. footnote text at 9pt, chapter heading at 22pt. This doesn't depend on the document main font size. LaTeX, years before, was only a bit better here, defining different relative sizes (but assuming a book is always typeset in 10pt, 11pt or 12pt). (3) Many commands, say book sectioning, rely on hard-coded vertical skips. But what if one wants a different vertical spacing? Two solutions come to mind, either redefining the relevant commands (say **\chapter**) or temporarily redefining the skips (say, **\bigskip**)… In a way, it all sounds very clumsy, cumbersome, somehow *ad hoc*, and… here, LaTeX-like. Which is not necessarily wrong (there is no offense intended here), but why not try a different approach?

Actually, this is what most modern word-processing software have been doing for a while, be it MS-Word, Libre/OpenOffice and cognates… They all introduce the concept of "styles", in actually three forms at least: character styles, paragraph styles and page styles. But also, frame styles, list styles, and table styles, to list a few others.

This package is an attempt at implementing such ideas, or a subset of them, in SILE.

Let's have a look at some recent version of LibreOffice…

Character styles include: font (family, style and weight, size, language, features), font effects (color, decoration e.g. underlining and strikethrough, case), position (superscript, subscript), rotation… (and plenty of smaller features, e.g. borders, which interest is a bit doubtful, heh!).

Paragraph styles, in addition to the above, include: indent and spacing, alignment, outline & numbering, text flow (notably, breaks).

Page styles include: page layout (a bit akin to SILE's masters), header and footer, folio numbering, background and such fancy things. Columns too, though this could be debatable.

— Anything below this point is still experimental, and likely unstable. —

To define a (character) style, one uses the following syntax (with any of the internal elements being optional):

\style:define[name=⟨*name*⟩**]{**
  \font[⟨*font specification*⟩**]**
  \color[color=⟨*color*⟩**]**
}

Can you guess how this *Style* was defined?
A style can also inherit from a previously defined style:

\style:define[name=⟨*name*⟩**, inherit=**⟨*other-name*⟩**]{**
  …
}

To apply a (character) style to some content, one just has to do:

\style:apply[name=⟨*name*⟩**]{**⟨*content*⟩**}**

Regarding re-definitions now, the first syntax below allows one to change the definition of style ⟨*name*⟩ to new ⟨*content*⟩, but saving the previous definition to ⟨*saved-name*⟩:

\style:redefine[name=⟨*name*⟩**, as=**⟨*saved-name*⟩**]{**⟨*content*⟩**}**

From now on, style \⟨*name*⟩ corresponds to the new definition, while \⟨*saved-name*⟩ corresponds to previous definition, whatever it was.

Another option is to add the **inherit** option to true, as show below:

\style:redefine[name=⟨*name*⟩**, as=**⟨*saved-name*⟩**, inherit=true]{**⟨*content*⟩**}**

From now on, style \⟨*name*⟩ corresponds to the new definition as above, but also inherits from \⟨*saved-name*⟩ — in other terms, both are applied. This allows one to only leverage the new definition, basing it on the older one.

Note that if invoked without ⟨*content*⟩, the redefinition will just define an alias to the current command (and in that case, obviously, the **inherit** flag is not supported). It is not clear whether there is an interesting use case for it (yet), but here you go:

\style:redefine[name=⟨*name*⟩**, as=**⟨*saved-name*⟩**]**

Finally, the following syntax allows one to restore style ⟨*name*⟩ to whatever was saved in ⟨*saved-name*⟩, and to clear the latter:

\style:redefine[name=⟨*name*⟩**, from=**⟨*saved-name*⟩**]**

So now on \⟨*name*⟩ is restored to whatever was saved, and \⟨*saved-name*⟩ is no longer defined.

The package also defines a **\style:font** command, which is basically the same as the standard **\font** command, but additionaly supports relative sizes with respect to the current **font.size**. It is actually the command used when applying a font style specification. For the sake of illustration, let's assume the following definitions:

```
\style:define[name=smaller]{\font[size=-1]}
\style:define[name=bigger]{\font[size=+1]}
\define[command=smaller]{\style:apply[name=smaller]{\process}}
\define[command=bigger]{\style:apply[name=bigger]{\process}}
```

Then:

```
Normal \smaller{Small \smaller{Tiny}},
Normal \bigger{Big \bigger{Great}}.
```

Yields: Normal Small Tiny, Normal Big Great.

Where do we go now? Paragraph and page styles haven't been proposed. Character styles could include other features. In other terms, this is an experimental work in progress!

1.6 redefine: (basic) command redefinition

The **redefine** package can be used to easily redefine a command under a new name.

Sometimes one may want to redefine a command (e.g. a font switching hook for some other command, etc.), but would also want to restore the initial command definition afterwards at some point, or to invoke the original definition from the newly redefined one.

This package is just some sort of quick "hack" in order to do it in an easy way. It is far from perfect, it likely has implications if users starts saving and restoring commands in a disordered way, but it can do the magic in some fairly reasonable symmetric cases.

The first syntax below allows one to change the definition of command ⟨*name*⟩ to new ⟨*content*⟩, but saving the previous definition to ⟨*saved-name*⟩:

\redefine[command=⟨*name*⟩**, as=**⟨*saved-name*⟩**]{**⟨*content*⟩**}**

From now on, invoking \⟨*name*⟩ will result in the new definition to be applied, while \⟨*save-name*⟩ will invoke the previous definition, whatever it was.

Of course, be sure to use a unique save name: otherwise, if overwriting an existing command, you will get a warning, at your own risks...

If invoked without ⟨*content*⟩, the redefinition will just define an alias to the current command:

\redefine[command=⟨*name*⟩**, as=**⟨*saved-name*⟩**]**

The following syntax allows one to restore command ⟨*name*⟩ to whatever was saved in ⟨*saved-name*⟩, and to clear the latter:

\redefine[command=⟨*name*⟩**, from=**⟨*saved-name*⟩**]**

So now on \⟨*name*⟩ is restored to whatever was saved, and \⟨*saved-name*⟩ is no longer defined. Again, if the saved name corresponds to some existing command in a broader scope, things may break.

1.7 teidict: XML TEI P4 print dictionaries

This package supports a subset of the (XML) TEI P4 “Print Dictionary” standard, as suitable for the Sindarin Dictionary project, and assumes a similar structure to the latter, see Data model.

The main pain point is that such a dictionary is a heavily “semantic” structured mark-up (i.e. a “lexical view”, encoding structure information such as part-of-speech etc. without much concern for its exact textual representation in print form), much more than a “presentational” mark-up. Some XML nodes may contain many things one needs to ignore (such as spaces, mostly) or supplement (such as punctuation, parentheses, numbering… and again, proper spaces where needed). Without XPath to check siblings, ascendants or descendants, it may become somewhat hard to get a nice automated output (and even with XPath, it is not that obvious). In other terms, the solution proposed here is somewhat *ad hoc* for a specific type of lexical TEI dictionary and depends quite a lot on its structural organization.

This package is not intended to be used as-is, but along with the **teibook** class, which loads it as well as a number of extra packages. It itself relies on a few settings that one would usually define in a preamble document, e.g.:

sile -I preambles/dict-sd-en-preamble.sil ⟨*dictionary.xml*⟩