An (experimental) **styles** package for SILE

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 could be debated). First, 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 **…:font**? Something looks amiss. Also, one ends up redefining a lot of such hooks, without any kind of inheritance between them. Secondly, many commands, say book sectioning, rely on hard-coded vertical skips. But what if one wants a different 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… 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 to implement 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 nd weight, size, language, features), font efects (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 footers, folio numbering, background and such fancy things. Column 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 \⟨*save-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 \⟨*save-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** flags 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 styme ⟨*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 (e.g. -1) 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!