Simon's Improved Layout Engine

SILE is a new text layout engine. It takes the best ideas from TeX and friends (pdftex, XeTeX and so on), adds some ideas of its own, and is implemented from the ground up using modern technologies. SILE's focus is on extensibility and clean design.

1 Installing

SILE requires working installations of Node.js, Pango and Cairo. Once you have these in place through your OS's package manager, you then need to install SILE's prerequisite modules:

```
> npm install node-pangocairo
> npm install .
```

Until SILE has its own working installer, you should manually copy the `sile` binary to `/usr/local/bin/`, and copy the `class`, `core`, `languages` and `packages` directories to `/usr/local/sile`. (Alternatively, you can set the `SILE_PATH` environment variable to the directory where you downloaded SILE.)

2 SILE Concepts

If you are familiar with TeX, most of SILE's concepts will be familar to you. There are a couple of new ideas:

Package. A SILE package is a JavaScript file which extends the way SILE operates. It may implement new commands, change the operation of the typesetter, and so on. This is distinct from a SILE file, which contains text to be typeset and basic macros.

Frame. Whereas TeX makes up pages entirely out of boxes and glue, SILE implements the *frame* as an additional abstraction between boxes and pages, and there can be multiple frames on a page. (There are two frames on this page: a content frame and a folio frame.) We will example frames in more detail later, but the basic concept should be familiar to anyone who has used graphical page layout tools such as InDesign or PageMaker.

3 A Basic SILE document

SILE documents are, by convention, XML files. *(They don't need to be; SILE's typesetting back-end component and its input front-end component know little about each other, and it's perfectly possible to write another front-end which reads files in more of a TeX-like style, if that's something you want to do. But the default front-end reads XML, and so that's what we'll describe in this manual.)*

Here is the most basic example of a SILE document:

```
 <sile papersize="a4">
Hello world.
</sile>
```

This will produce an A4 document with the text "Hello world" at the top, and the page number centered at the bottom. The `papersize` attribute is required; SILE knows about a number of standard paper sizes, but if you wish to use a non-standard size, you can specify the page size as width-by-height dimensions like so: `papersize="129mm x 198mm"`. *(SILE understands dimensions specified in mm, cm, in, pt or px. When the paper size is established, you can also specify dimensions as a percentage of the page size; horizontal dimensions will be understood as a percentage of the page width, and vertical dimensions as a percentage of the page height.)*

SILE has a number of built-in commands which are available as XML tags.

3.1 Including things

You can include another SILE XML file using the `<include>` tag, like so:

```
 <sile papersize="a4">
Hello world.
<include src="chapter1.sil"/>
<include src="chapter2.sil"/>
</sile>
```

Or you can include a SILE package using the `<script>` tag:

```
 <sile papersize="a4">
<script src="packages/image" />
Hello world.
<img src="smiley.png" width="100px" height="100px"/>
</sile>
```

The brave and foolhardy can include raw Javascript code within a `<script>` tag in a way that should be familiar to HTML programmers:

```
 <sile papersize="a4">
Hello world.
<script>console.log("Hello Javascript");</script>
</sile>
```

However, it is recommended that any serious scripts go in their own package file.

3.2 Styling things

The basic way to change the way text appears is using the `<font>` tag. This tag takes the following attributes:

`family`: Specifies the font family. This can be done in a CSS-like "stack", a comma-separated list. SILE will use the first font it can find which provides the characters you are trying to typeset.

`size`: The font size in points.

`weight`: The font weight in CSS numeric format.

`style`: The font slant style ("oblique", "italic" or "normal").

You may provide as many or as little of these attributes as required. The tag can be used in two ways. An empty tag changes the parameter permanently until the next tag:

```
 <font size="24" weight="700"/>
I am shouting here!
<font size="12" weight="400"/>
I am not shouting any more.
```

Whereas a tag with content will only change the parameter for the content within the tag, restoring the previous font style afterwards:

```
SILE is <font style="italic">very</style> easy to use!
```

Of course, it would be boring to have to type `<font style="italic"` all the time, but we will improve that later.

3.3 Moving things

To round out the list of built-in commands, advanced users can typeset particular nodes directly, using the `<penalty>`, `<glue>`, and `<skip>` tags:

```
 <penalty penalty="500" flagged="1"/>
<glue width="0" shrink="0" stretch="10000"/>
<skip height="2pt" shrink="0" stretch="2pt"/>
```

So, for instance, you can center a line by saying:

```
<glue stretch="10000"/>Hello, world<glue stretch="10000"/>
```

3.4 Macros

However, this gets cumbersome fast. Thankfully, SILE allows you a basic way to easily define your own commands. The last built-in tag you need to know about is the `<define>` tag. For instance, this document includes a macros file which starts with the following defines:

```
 <define command="center"><glue stretch="10000"/><process/><glue stretch="10000"/></define>
<define command="em"><font style="italic"><process/></font></define>
<define command="sectionsfont">
 <font family="Frutiger LT Std" weight="700"><process/></font>
</define>
```

As you can see, we define a new command with the `<define>` tag, giving the command name as an attribute. When this command is used, the content inside the `<define>` tag will be processed as though it appeared in that location of the document file. Additionally, the `<process>` instruction instructs SILE to process the contents of the command tag. At risk of belabouring the obvious, this means that:

```
SILE is <em>very</em> easy to use!
```

is now precisely equivalent to

```
SILE is <font style="italic">very</font> easy to use!
```

Macros cannot take any parameters or attributes to modify their activity other than <process>ing their contents. If you want parameters, define a command using a package. Macros can, however, contain other macros, and this gives a certain degree of expressibility:

```
 <define command="headline">
<skip height="1cm"/>
<sectionsfont><center><font size="20"><process/></font></center></sectionsfont>
<skip height="0.5cm"/>
</define>
```

4 Doing more

4.1 Frame and flow

4.2 Using SILE packages

5 SILE hacking