

Page Design in \LaTeX 3

Morten Høgholm

\LaTeX 3 Project

TUG 2006 9–11 November 2006

Outline

- 1 State of Affairs
- 2 Page Design
- 3 Presentation of the Tools
- 4 Outlook

What the $\text{\LaTeX}3$ Project does

- Maintaining \LaTeX :
 - Fixing bugs.
 - Listening to feature requests for the tools packages.
 - Checking that one fix doesn't break other things!
- Talking to engine programmers.
- Explore (and improving) new technologies.
- Deciding what should go in the $\text{\LaTeX}3$ kernel.
- Adding code to the $\text{\LaTeX}3$ kernel.

What the $\text{\LaTeX}3$ Project does

- Maintaining \LaTeX :
 - Fixing bugs.
 - Listening to feature requests for the tools packages.
 - Checking that one fix doesn't break other things!
- Talking to engine programmers.
- Explore (and improving) new technologies.
- Deciding what should go in the $\text{\LaTeX}3$ kernel.
- Adding code to the $\text{\LaTeX}3$ kernel.
- Deciding on a release date. . .

The $\text{\LaTeX}3$ kernel

$\text{\LaTeX}3$ has three layers:

- 1 The core programming tools `expl3`.
- 2 An intermediate layer with a designer interface using templates and the core API.
- 3 At the document level there is the document syntax, which relates only to the second layer. It does not use the core API.

The experimental code

The experimental code is divided into two areas:

- The expl3 code.
- The xpackages built on expl3 code

Both parts work on top of \LaTeX as packages.

All this code is available from a publicly available SVN repository at `http:`

`//www.latex-project.org/svnroot/experimental/trunk/`.

The expl3 code

The expl3 code is (read: was) an experimental coding scheme.

- Originally presented in 1997.
- A coding scheme trying to make T_EX look like a *real* programming language!
- Spurious spaces don't exist.
- No interwoven `\expandafter`, `\csname` and `\endcsname`.
- Several different data types with mutator/accessor functions, mapping functions, etc.
- Major revision of the code started in late 2004. Fairly stable now.

The xpackages

Higher-level modules using the expl3 API.

templates parameterized generic functions.

xparse Provides a standard L^AT_EX syntax to interface with templates.

xor The new output routine.

galley Gaining control of everything that is supposed to go on the galley. This means no more problems with `\specials`, `\writes` cramping our style!

Others See the repository for more...

Small xparse example

The standard \LaTeX document level syntax is covered by the `xparse` package.

A small example of how to define a command with optional star and optional argument:

Small xparse example

The standard \LaTeX document level syntax is covered by the `xparse` package.

A small example of how to define a command with optional star and optional argument:

```
\DeclareDocumentCommand\testcmd{som}{  
  \IfBooleanT{#1}{Star}  
  \IfValueTF{#2}{#2}{NoValue}  
  ‘‘#3’’% mandatory argument  
}
```

Outline

- 1 State of Affairs
- 2 Page Design**
- 3 Presentation of the Tools
- 4 Outlook

Identifying the Problem

All pages are not identical.

- Left and right margins usually change every page.
- Rotated pages including margins.
- Float pages, page spreads.
- For some designs the recto and verso pages have different layouts.
- With online PDF-documents, the page size may even change mid-document!

Current solution in \LaTeX

The standard \LaTeX interface is characterized by the following:

- Right and left pages are identical except for `\evensidemargin` and `\oddsidemargin`
- All parameters are global.
- If a temporary change of parameters is desired:
 - Store values elsewhere, or
 - Change these global parameters locally in a group. . .
- Standard \LaTeX interface is no interface: Change parameters individually and keep your fingers crossed.
- Alternatively use the layout package to look at the outcome.

Current solution in \LaTeX , cont.

Going beyond the \LaTeX kernel we find better tools.

- Some packages and document classes make it easier
 - The geometry and typearea packages provide two different interfaces.
 - The memoir document class provides a third interface.
- All are very useful and give the user/designer complete control of parameters and check the parameter values.
- However, all of these just translate to the basic \LaTeX commands. They do not solve the problems outlined above.

Working Out a Solution

We will do things a little differently.

- Treat each type of page as its own entity. Hence there are different types:
 - The usual recto and verso pages
 - Spread pages both recto and verso parts
 - Rotated pages.
 - And others. . . Designers are imaginative!
- This means every page type has its own set of parameters.
- The parameters must be easily restored and retrieved.
- The shipout routine is used to switch between sets of parameters.

Outline

- 1 State of Affairs
- 2 Page Design
- 3 Presentation of the Tools**
- 4 Outlook

Property lists

A property list is a list data type with the following structure:

```
\⟨key 1⟩{info 1}  
\⟨key 2⟩{info 2} ...
```

They have mutator and accessor functions in the core API.

A small example:

Property lists

A property list is a list data type with the following structure:

```
\⟨key 1⟩{info 1}  
\⟨key 2⟩{info 2} ...
```

They have mutator and accessor functions in the core API.

A small example:

```
\prop_new:N \g_TUGmmvi_plist  
\prop_gput:NNn \g_TUGmmvi_plist \firstname{Morten}  
\prop_gput:NNn \g_TUGmmvi_plist \surname{H\o gholm}  
...
```

Property lists, cont.

A few more examples showing the abilities of the core API

Property lists, cont.

A few more examples showing the abilities of the core API

```
\toks_set:Nn \l_tmpa_toks{Morten}
\prop_gput:cNo {g_TUGmmvi_plist}
                \firstname{\toks_use:N \l_tmpa_toks}
\prop_gput:ccn {g_TUGmmvi_plist} {surname}{H\o gholm}
```

Property lists, cont.

A few more examples showing the abilities of the core API

```
\toks_set:Nn \l_tmpa_toks{Morten}
\prop_gput:cNo {g_TUGmmvi_plist}
                \firstname{\toks_use:N \l_tmpa_toks}
\prop_gput:ccn {g_TUGmmvi_plist} {surname}{H\o gholm}
```

An example of mapping a function on a property list:

Property lists, cont.

A few more examples showing the abilities of the core API

```
\toks_set:Nn \l_tmpa_toks{Morten}
\prop_gput:cNo {g_TUGmmvi_plist}
    \firstname{\toks_use:N \l_tmpa_toks}
\prop_gput:ccn {g_TUGmmvi_plist} {surname}{H\o gholm}
```

An example of mapping a function on a property list:

```
\def:NNn \testcmd:Nn 2 {
  \io_put_term:x {
    \token_to_string:N #1 :~ \tlist_to_string:n {#2}
  } }
\prop_map_function:NN \g_TUGmmvi_plist \testcmd:Nn
```

Result is “\firstname: Morten \surname: Høgholm”

Using property lists for page parameters

The idea:

- For each page type, create a property list and put each parameter into it.
- These property lists will have a standard form of

```
\l_page_recto_stockheight_dim {\l_page_recto_stockheight_dim}  
\l_page_recto_stockwidth_dim {\l_page_recto_stockwidth_dim}  
...
```

Here `recto` is the page type.

- Using the mapping function shown earlier and appropriate auxiliary function, all values can be stored in a macro in a simple form:

```
\dim_set:Nn \l_page_recto_stockheight_dim {500pt}  
\dim_set:Nn \l_page_recto_stockwidth_dim {300pt}  
...
```

Templates

- Templates are parameterized, generic functions.
- Templates have a standard syntax.
- No direct connection with document syntax: only takes regular arguments.
- Demo

A template interface for page layouts

- 1 The template sets parameters for a standard page.
- 2 These parameters are set for the page type at hand.
- 3 An auxiliary function constructs a macro for setting the values for all parameters.
- 4 At the shipout, \LaTeX copies the dimensions in the macro to parameters used by the kernel.

A template interface for page layouts

The template call takes one argument: The name for the page type.

```
\DeclareTemplateType{pagelayout}{1}
\DeclareTemplate{pagelayout}{simple}{1}{
  textheight  = 1 [8in]  \g_page_std_textheight_dim ,
  textwidth   = 1 [6.5in] \g_page_std_textwidth_dim ,
}{
  \DoParameterAssignments
  \dim_set:cn {l_page_#1_textheight_dim}{\g_page_std_textheight_dim}
  \dim_set:cn {l_page_#1_textwidth_dim}{\g_page_std_textwidth_dim}
  \page_store_page_dimensions:n {#1}
}
```

A template interface for page layouts

The template call takes one argument: The name for the page type.

```
\DeclareTemplateType{pagelayout}{1}
\DeclareTemplate{pagelayout}{simple}{1}{
  textheight = 1 [8in] \g_page_std_textheight_dim ,
  textwidth  = 1 [6.5in] \g_page_std_textwidth_dim ,
}{
  \DoParameterAssignments
  \dim_set:cn {l_page_#1_textheight_dim}{\g_page_std_textheight_dim}
  \dim_set:cn {l_page_#1_textwidth_dim}{\g_page_std_textwidth_dim}
  \page_store_page_dimensions:n {#1}
}
```

Now running the template:

```
\UseTemplate{pagelayout}{simple}
  { textheight= 9in } {simple-recto}
\UseTemplate{pagelayout}{simple}{}{simple-verso}
\UsePageLayout{2}{simple-recto,simple-verso}
```

More advanced templates

More advanced templates can be constructed:

- Checking if parameters have legal values.
 - Easy task due to property list tools.
- Auto-completion.

Outline

- 1 State of Affairs
- 2 Page Design
- 3 Presentation of the Tools
- 4 Outlook**

To do

Some things need to be done:

- Make prototype work with xor.
- Produce more templates.

To do

Some things need to be done:

- Make prototype work with `xor`.
- Produce more templates.
- All the other stuff we need to do for $\text{\LaTeX}3$.