tbook

a system for XML authoring
version 1.5.2

Torsten Bronger

This manual is for **t**book (version 1.5.2), which is an XML file format and an authoring tool.

Table of Contents

1 Introduction **1**

Features 1
Availability 2

2 Installation **3**

2.1 LaTeX packages needed by tbook 3
2.2 Linux RPM installation 3
2.3 Linux installation from the sources 3
2.3.1 Prerequisites 4
2.3.2 The installation with make 4
2.3.3 Installation process from sources in detail 5
2.4 Windows installation with the installer 6
2.5 Windows installation from the sources 6

3 Usage **9**

3.1 Where to look something up 9
3.2 Starting a new document 9
3.3 The shell scripts 9
3.3.1 Invoking the shell scripts 9
3.3.2 The "make" scripts 10
3.3.3 Optional argument for the shell scripts 10
3.3.4 Shell script dependencies 11
3.3.5 Full example with the scripts 11
3.4 Your original graphics files 13
3.5 Viewing your document in a browser 13
3.5.1 Possible HTML variants with tbook 13
3.5.2 XHTML pages 13
3.5.3 The Universal Math Stylesheet 14
3.6 Jade/nsgmls issues 14
3.7 Other XSLT processors: Saxon 7, Xalan, ... 15

4 Configurability **17**

4.1 XSLT parameters 17
4.1.1 Further XSLT parameters 18
4.2 CSS style sheets 20
4.3 LaTeX `cfg` file 21
4.4 LaTeX `sty` file 22
4.5 Driver files 22

5 tbook's "almost LaTeX" formula syntax . . . 25
5.1 MathML v. simplified LaTeX . . . 25
5.2 MathML in tbook . . . 26
5.2.1 Equation arrays . . . 26
5.2.2 Equation numbers . . . 27
5.2.3 Under- and overbraces . . . 28

6 Human language support . . . 29

7 The `style` and `class` attributes . . . 31
7.1 Text colour . . . 31

8 Bibliography . . . 33

9 Elements reference . . . 35
9.1 Minimal example . . . 35
9.2 Terms and symbols in the element reference . . . 35
9.3 Top level elements . . . 36
9.4 Parts, chapters and sections . . . 40
9.5 Paragraphs & friends . . . 43
9.6 Font change . . . 45
9.7 Cross references . . . 46
9.8 LaTeX-like mathematics . . . 49
9.9 Theorems and proofs . . . 51
9.10 Miscellaneous . . . 52
9.11 Quoted and verbatim material . . . 55
9.12 Lists . . . 59
9.13 "Floats" and their contents . . . 61
9.14 Tables . . . 66
9.15 Elements of the frontmatter . . . 68
9.16 References . . . 72
9.17 Index . . . 73
9.18 letter elements . . . 75

10 Known problems with tbook and associated applications . . . 77
10.1 Associated applications . . . 77
10.1.1 Saxon 6.5.2 . . . 77
10.1.2 Netscape/Mozilla . . . 77
10.1.3 Internet Explorer and MathPlayer . . . 78
10.1.4 Konqueror and Safari . . . 78
10.1.5 Jade . . . 79
10.1.6 HTML validators . . . 79
10.2 List of wishes for tbook . . . 79

11 tbook source code documentation 81
11.1 The DTX files 81
11.2 CWEB files 81
11.3 If you can read German 81

Index 83

1 Introduction

Authors of scientific or technical texts often experience the following two problems:

1. They want to focus on contents, i. e. the actual ideas and a pleasant language, but they feel forced to take care of totally different things like page margins, line skips, fonts, table layout etc.
2. They want to (or must) have the end product in different formats, e. g. PDF, RTF (for wordprocessors), and if possible HTML.

Authors of non-technical texts may face very similar problems.

The solution is to use XML. XML is a generic file format that has to be adapted for special purposes. **t**book is one of these XML applications. It was inspired by LaTeX and can be used for documents typically written with it, although you don't need to know about LaTeX in order to use **t**book.

tbook solves the above problems:

1. **t**book documents do not contain any layout information. They consist only of textual contents and structure. Stylesheets that are automatically applied later turn the document into a pleasant output.
2. Theoretically, **t**book files can be converted to virtually all other document formats. The conversions are lossless and dependable. Practically, complete conversions to Postscript, PDF, HTML, and DocBook already exist.

   Postscript and PDF are generated via LaTeX which allows for very good layout quality. DocBook can be converted into fairly good RTFs that can be read by most wordprocessors.

Some people may dislike to lose control over the layout. These people can customise **t**book's behaviour in many ways. However, the strict separation of contents and layout always remains intact.

Another difficulty is that XML files are text files, and they look rather intimidating at first sight. There are only few good XML editors available so far. However, some of them help you greatly nevertheless, e. g. GNU Emacs and Cooktop[1] and new editing systems come into existence almost every week.

Features

This is what **t**book can do so far:

- An XML DTD that is suitable for demanding, especially scientific, texts, but it is also as simple and small as possible, and uses similar names as LaTeX. Supported text classes: book, article, and letter.
- It produces HTML, XHTML+MathML, Postscript, PDF, and DocBook output.
- It works with formulae, graphics, tables, (all three with numbering) bibliography, and index. Tools that have proven their efficiency with LaTeX (BibTeX, xindy etc.) are also usable with **t**book.

[1] Windows only

- The print output caters high typographic demands, using high level LaTeX typesetting mechanisms. This includes neat graphics and their labelling with Psfrag. Additionally, it produces small PDFs.
- It is easily configurable. For LaTeX via a user package file, for HTML via a CSS fragment, and there are yet other ways.
- As much as possible is done automatically, e. g. via shell scripts that are synchronised with the document continually.
- It works with different languages and allows for direct unicode input. Supported languages so far: English, German, French, Italian, Spanish, Catalan, Danish, Dutch, Polish, Swedish, Norwegian, Portuguese, Brazilian, Finnish, and Russian.
- It is possible to input XML code directly without being killed by tons of characters. (That counts twice for formulae.)
- It's a real life tool, no academic project. If it doesn't work properly for you, complain!

Availability

Project's homepage
`http://tbookdtd.sourceforge.net/`

Download area, bug reports, mailing list, CVS tree, and more
`http://sourceforge.net/projects/tbookdtd`

tbook DTD reference
`http://tbookdtd.sourceforge.net/dtd/`

FAQ `http://tbookdtd.sourceforge.net/tb-faq.pdf`

2 Installation

The installation is not trivial, mainly because tbook desperately needs some other applications, that are not included in the distribution. The most important ones are LaTeX, Ghostscript and Saxon.

However, tbook's homepage offers Linux RPMs for tbook and Saxon, and for Windows, the homepage offers a mere .EXE file that reduces the installation effort drastically.

2.1 LaTeX packages needed by tbook

tbook needs many LaTeX packages. Those that are part of the LaTeX core (`babel`, `fontenc`, `array`, etc) are nothing to worry about, every non-ancient LaTeX has them. The other packages are: `booktabs`, `caption2`, `fancyhdr`, `geometry`, `helvet`, `hypcap`, `pdfcolmk`, `hyperref`, `marvosym`, `mathptmx`, `minitoc`, `natbib`, `nicefrac`, `pifont`, `psfrag`, `ragged2e`, `relsize`, `sectsty`, `soul`, `url`, `wasysym`. That's certainly a big part of the crème de la crème of LaTeX packages. Be aware that `helvet`, `mathptmx`, and `hyperref` must be as new as possible. If your versions are too old, LaTeX will gently complain.

Every modern LaTeX distribution contains most of them, maybe all; if your LaTeX distribution let you choose between different options, you should prefer a 'complete' or 'full' installation, else LaTeX itself will complain and tell you which package you have to install. You will find it on every CTAN server, e. g. `http://www.tex.ac.uk/tex-archive/macros/latex/contrib/`.

Your Babel system must know the human languages you want to use. Shouldn't be very surprising, and normally is no problem.

If you have any trouble, contact the current maintainer of tbook (the author of this document).

2.2 Linux RPM installation

The installation of the tbook RPM is not different from any other RPM installation: Download the file, login as root, and then input e. g.

```
rpm -i tbook-1.5.2-1tb.i386.rpm
```

However, tbook depends on other programs and it would have been very inconvenient if I had included those dependencies within the RPM system. Only one dependency is explicit: Saxon. But you get an RPM version of Saxon from tbook's homepage, too (`http://sourceforge.net/projects/tbookdtd`). Saxon must be installed before you install tbook. Saxon itself needs Java, at least version 1.1, although JDK 1.3 makes Saxon run *much* faster. *Warning*: You can't use tbook with Saxon 7!

On tbook's homepage you also get an RPM version of xindy.

2.3 Linux installation from the sources

There are two possibilities to install tbook under Linux: The usual source distribution that is explained here, and the RPM installation described in the previous section. The RPM way is much simpler, however, it may not work for you; in this case you have to build tbook from the sources.

2.3.1 Prerequisites

The following programs must be installed in order to run tbook:

- LaTeX, pdfLaTeX, (both new and well equipped, see Section 2.1 [Needed LaTeX packages], page 3), dvips,[1]
- Ghostscript[2], ps2pdf (part of Ghostscript),
- pnmtopng, ppmtojpeg, pnmfile,[3] (not *really* vital),
- xindy[4] if you need indexing,
- jpeg2ps[5] and
- Saxon 6.5.x[6] (I don't support Saxon 7 yet).

Much of this may be already installed with your system.

For the Saxon you have to provide a shell script named '`saxon`' that calls it. It may look like this:

```
CLASSPATH=~/jars/saxon.jar:$CLASSPATH
export CLASSPATH
java -ms15000000 com.icl.saxon.StyleSheet $1 $2 $3 $4 $5 $6 $7 $8 $9
```

However, newer versions of Saxon (newer than 6.5.2) may need another class name. See the Saxon homepage at "installation"–"changes" for further details.

In order to install tbook with some convenience, you also need CWEB[7], sed, patch and Flex[8]. However, the Linux/Intel binaries are included in the distribution, as well as all BibTeX-Styles, if you have trouble to build them.

2.3.2 The installation with make

If all preconditions are met, unpack the tar ball. If you wish, you can check the integrity of all files with non-ASCII characters with

```
md5sum -c tbook.md5
```

Adjust some paths in the Makefile, run '`make`' and then (as root) `make install`. You can pass `LOCAL=...` as an option to '`make`' and change the root of the installation. It's default value is '`local/`'. In particular, you can set it to the empty string.

After that, you will probably have to update TeX's file database, which depends on your TeX implementation. The program that does this may be called '`texhash`' or '`mktexlsr`'.

[1] all of them at `http://www.tex.ac.uk/`

[2] `http://www.cs.wisc.edu/~ghost/doc/AFPL/get704.htm` ; *don't* use GNU Ghostscript

[3] all three at `http://sourceforge.net/netpbm`

[4] `http://sourceforge.net/projects/xindy`, or as an RPM from tbook's project page at `http://sourceforge.net/projects/tbookdtd`

[5] On CTAN at '`nonfree/support/jpeg2ps/`'

[6] `http://sourceforge.net/projects/saxon`, or as an RPM from tbook's project page at `http://sourceforge.net/projects/tbookdtd`

[7] `http://www-cs-faculty.stanford.edu/~knuth/cweb.html`

[8] sed, patch and Flex are part of the GNU project at `http://www.gnu.org`

2.3.3 Installation process from sources in detail

In the following I describe what happens exactly when running `make install` (unless, of course, you've changed some paths, which you probably will have to). You may read this only if you are really keen to know it.

1. Unpacks all files contained in the DTX files via docstrip.
2. Creates the BibTeX style files using the '`custom-bib`' package[9] and copies them to '`/usr/local/teTeX/share/texmf/bibtex/bst/tbook`'.
   - The files '`tb??l.bst`' (with '`??`' being the language) are for LaTeX output.
   - The files '`tb??h.bst`' (with '`??`' being the language) are for HTML output.
3. Builds the binaries '`tbrplent`', '`tbcrent`', '`tb2xindy`' and '`bibfix`' from the sources.
   - '`tbrplent`' is a filter program that scans for non-ASCII UTF-8s in the input stream and creates decent LaTeX macros or, if possible, Latin-1 characters for the output stream.
   - '`tbcrent`' is a helper that creates the resource file for '`tbrplent`', starting from XML entity files. Maybe you'll never need to run it.
   - '`tb2xindy`' is a scanner to teach xindy digesting XML input.
   - '`bibfix`' is a filter that transforms BibTeX's XML output into (hopefully) real XML.
4. Copies these files into '`/usr/local/bin`'.
5. Copies the four shell scripts '`tbtolatex`', '`tbtohtml`', '`tbtodocbk`', and '`tbprepare`' into '`/usr/local/bin`', too.
   - '`tbtolatex`' takes a tbook XML file name (without the '`.xml`' extension) and creates via Saxon and '`tbrplent`' a LaTeX input file.
   - '`tbtohtml`' does the same thing for HTML output.
   - '`tbtodocbk`' does the same thing for DocBook 4.2 XML output.
   - '`tbprepare`' is used *once*, when you start a new document. It copies needed files into the current directory and creates a template tbook XML file to start with.
6. Creates the directories '`/usr/local/teTeX/share/texmf/tex/latex/tbook`' and '`/usr/local/lib/tbook`'.
7. Copies the LaTeX input files '`t1pcrv.fd`', '`tbook.sty`' and '`tbook-pl.sty`' into '`.../texmf/tex/latex/tbook`'.
   - '`t1pcrv.fd`' and '`ts1pcrv.fd`' are an alternative version of Adobe Courier, because it's just too big.
   - '`tbook.sty`' is the mandatory package for all generated LaTeX files.
   - '`tbook-pl.sty`' ("tbook plain") is an example style package for generated LaTeX files.
8. Copies the tbook XML files '`tbook.dtd`', '`tblatex.xsl`', '`tbhtml.xsl`', '`tbdocbk.xsl`', '`tbtoltx.xsl`', '`tbtohtml.xsl`', '`tbtohtml4.xsl`', '`tbtodb.xsl`', '`tbcommon.xsl`', '`hmml2dst.dtd`', '`tbents.txt`' and '`tbtmplte.xml`' to '`/usr/local/lib/tbook`'.
   - '`tbook.dtd`' is the tbook XML DTD.

[9] If you don't have '`custom-bib`', use the prepared `bst` files in the `bst/` directory.

- '`tblatex.xsl`' is the XSLT for **t**book ⇒ LaTeX.
- '`tbhtml.xsl`' is the XSLT for **t**book ⇒ XHTML.
- '`tbdocbk.xsl`' is the XSLT for **t**book ⇒ XML DocBook.
- '`tbtoltx.xsl`', '`tbtohtml.xsl`', '`tbtohtml4.xsl`', and '`tbtodb.xsl`' are so called driver files for the above two stylesheets, see below "Configurability".
- '`tbcommon.xsl`' contains shared code for '`tbhtml.xsl`' and '`tblatex.xsl`'.
- '`hmml2dst.dtd`' contains an awful hotchpotch of MathML2's DTD, it's entities and HTML's entities. It's only needed as long as XML tools have no catalog feature.
- '`tbent.txt`' is created by '`tbcrent`' and needed by '`tbrplent`'. It contains the Unicode ⇒ LaTeX mappings.
- '`tbtmplte.xml`' is used as a template XML file for the launch of a new document project.

9. Copies the xindy style files '`tblatex.xdy`', '`tbhtml.xdy`' and '`tbook.xdy`' to '`/usr/local/lib/xindy/modules/misc`'.
   - '`tblatex.xdy`' is the xindy style file for a LaTeX index.
   - '`tbhtml.xdy`' is the xindy style file for an HTML index.
   - '`tbook.xdy`' contains shared commands for both index outputs.
10. Copies the manpages to '`/usr/local/share/man/`'.
11. Copies the Info file to '`/usr/local/share/info/`'.
12. Copies the DTD in browsable HTML format to '`/usr/local/share/doc/tbook`'.

2.4 Windows installation with the installer

There are also two methods to install **t**book on the Windows operating system. The first one that should be prefered is to use the NSIS based installer/uninstaller program. Basically it's just an .EXE file called '`tbook-1.5.2.exe`' that you have to start. But before you do that, please install:

1. TeX. Get the latest MikTeX distribution or the TeXLive CD. The installation shouldn't be too hard, but be careful that no package is missing (see Section 2.1 [Needed LaTeX packages], page 3).
2. Ghostscript. Get AFPL Ghostcript and install also GSView, although only Ghostscript is needed by **t**book.

The **t**book installer will also install Saxon 6.5.2 and xindy 2.0 for you.

This way has another advantage: A program called '`uninst.exe`' will uninstall **t**book, delete the entries in '`autoexec.bat`' and in the registry. To uninstall **t**book, you may also use the standard way via system control⇒software.

2.5 Windows installation from the sources

This is an emergency solution in case that you can't (or don't want to) install **t**book with the installer described in the previous section.

Make sure that you have installed the following programs on your system:

1. Saxon. You may find it at `http://saxon.sourceforge.net`. There are two versions: “Instant Saxon” which is easier to install but runs rather slowly and “Full Saxon” that comes as a Java archive. The Saxon homepage offers detailed installation instructions. Anyway, it’s important that there is an executable program called `saxon` on your computer, which may be an EXE or a batch file (shell script). *Don’t install Saxon 7, but Saxon 6.5.x!*
2. TEX. Get the latest MikTeX distribution or the TEXLive CD. The installation shouldn’t be too hard.
3. Ghostscript. Get AFPL Ghostcript and install also GSView, although only Ghostscript is needed by tbook.
4. xindy. This is only necessary if you want to create index directories for your texts. It normally doesn’t come with TEX, although it should. You find it on the xindy project page. Follow the installation instructions in the package, then installation should be easy.

Make sure that all programs are in your `PATH` environment variable and thus can be found by tbook!

Then install tbook: Copy the Zip file on your hard disk and unpack it with e. g. Winzip. All files are unpacked into a subdirectory of the current one. Open a command line interpreter (MS-DOS shell) and go into this subdirectory.

The first thing you have to do is to edit the file ‘`install.bat`’. Edit the `SET ...=...` entries at the beginning of the file and substitute your directory names for the default values. Then delete the two lines

```
pause
goto end
```

at the beginning of ‘`install.bat`’.

Then start ‘`install.bat`’. Maybe you have to be administrator to do this. After this, set the environment variable `TBLIBDIR` and set it to the value that you also gave in ‘`install.bat`’. For this, you probably have to edit ‘`autoexec.bat`’. At least on my system.

Make sure that the binary files of tbook wander in a directory (`BINDIR`) that is part of your `PATH`!

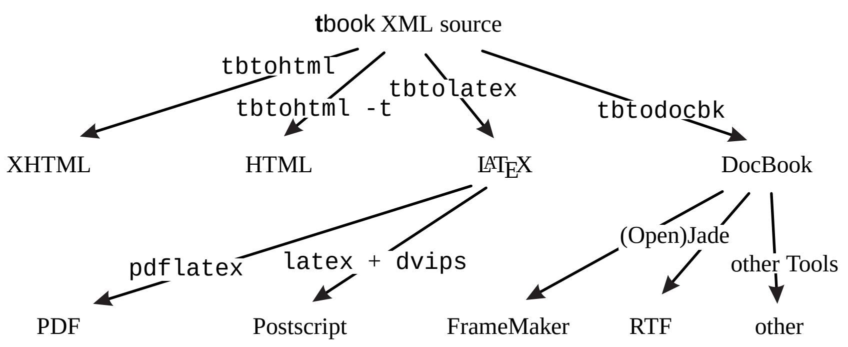
After that, you will probably have to update TEX’s file database. How this is done depends on your TEX implementation. With MikTEX e. g., it’s an item in the MikTEX program group of the start bar.

That’s it. Hopefully.

If you’re looking for a good free XML editor, have a look at Cooktop.

3 Usage

An overview:



3.1 Where to look something up

First some words about where to get help. The text you are reading is available in Info format with

```
info tbook
```

in HTML (‘`tbookman.html`’) in ‘`/usr/share/doc/tbook/`’, and in PDF format (‘`tbookman.pdf`’) (e. g. on tbook’s homepage). There are manpages available for ‘`tbtolatex`’, ‘`tbtohtml`’, ‘`tbtodocbk`’, ‘`tbprepare`’, and ‘`tbook`’ itself.

A browsable DTD is stored in the ‘`/usr/share/doc/tbook/`’ directory, which may be found at a different place depending on the options you chose during installation. But it’s also accessible online at `http://tbookdtd.sourceforge.net/dtd/`.

If all else fails, use the forums on tbook’s homepage. See [Web addresses], page 2.

3.2 Starting a new document

If you want to start a new text project, enter a fresh directory and type

```
tbprepare <dn>
```

(The real document name must be substituted for ‘`<dn>`’.) This copies some ‘`make??`’ shell scripts and a file called ‘`<dn>.xml`’ in the current directory. Then you may edit this XML file and enter your contents. Emacs does a great job here.

3.3 The shell scripts

From the user’s point of view, I now explain the most important part of tbook. It’s about the programs that transform any tbook document into something else. This is obviously vital if you want to use tbook in a productive way.

3.3.1 Invoking the shell scripts

You have the following commands available:

- `tbtolatex <dn> [<parameters>]` creates a LaTeX file ‘`<dn>.tex`’ ready to be processed by LaTeX or pdfLaTeX.

- `tbtohtml <dn> [<parameters>]` creates a file '`<dn>.xhtml`' with an XHTML version of your document.
- `tbtodocbk <dn> [<parameters>]` creates a file '`<dn>-db.xml`' with a DocBook version of your document.

The scripts '`tbtolatex`' and '`tbtohtml`' accept '`-t`' as the very first argument. Then they will create not an XHTML file, but an HTML 4 file with bitmap equations and greater CSS compatibility for older browsers.

3.3.2 The "make" scripts

Additionally, you can call the following "make" scripts whenever you think it's appropriate. They will help you very much: For example, they prepare your graphics automatically for the different output formats. But they are also responsible for index and bibliography. They are constantly kept up-to-date with the document.

> Please note that on some Unix/Linux installations you have to call the "make" scripts with `./` – so you have to input `./makeepss` instead of `makeepss`.

- '`makeepss`' creates EPS versions of all present JPEGs; well, of all that are included in your document. For dvips.
- '`makeidx`' creates an index for your output. For which output depends on which '`tbto??`' transformer you've called last.
- '`makebib`' creates a references list for your output. For which output depends on which '`tbto??`' transformer you've called last.
- '`makepdfs`' creates PDF versions of all included EPS'es. Psfrag elements are properly handled. For pdfLaTeX.
- '`makewebs`' creates JPEG or PNG versions of all included graphics, for HTML output.
- '`makeeqns`' creates PNG versions of all included formulae, for HTML output. See also the XSLT parameter '`preview-latex`' which can make this process *much* faster, see Section 4.1 [XSLT parameters], page 17.
- '`makepage`' creates a subdirectory with all files needed by the HTML version, i. e. the HTML file itself and the PNG/JPEGs.
- '`makeclean`' erases almost all generated and superfluous files. It uses no wildcards, for safety.

(Most of theses files are updated by a call of '`tbtolatex`', only '`makeidx`' and '`makebib`' are also updated by '`tbtohtml`'. '`tbtodocbk`' only updates '`makebib`'.)

Besides that, you will have to call LaTeX, pdfLaTeX and dvips as necessary.

3.3.3 Optional argument for the shell scripts

'`makeepss`', '`makepdfs`', and '`makewebs`' process by default all graphics in your document. This may be annoying, because it usually consumes a long time and you only want to re-create *one* graphics. Therefore they accept an optional argument which is the name of a graphics. If it's given, they process only *this* graphics, as in

```
makewebs niceimage
```

This (re-)creates only the web version of the graphics that is included with

```
<graphics file="niceimage"/>
```

in the tbook document. Similarly you can give '`eqn-<number>`' to '`makeeqns`':

```
makeeqns eqn-151
```

You can find out the number of the equation that you want to have re-created by using your browser to see which bitmap filename the formula has.

3.3.4 Shell script dependencies

The dependencies of the '`make??`' shell scripts and the '`tbto??`' transformation programs are somewhat subtle. In general, if you've changed your XML file significantly (e. g., you've added a new graphics), you have to call '`tbtolatex`' in any case, because only then all make files are up to date.

Then you should call '`makeepss`', '`makepdfs`', and '`makewebs`' with the name of the graphics as argument.

Then you can do the transformation to HTML or DocBook, or call (pdf)LaTeX.

If you want to update your index or bibliography, call the transformation to the desired format, call '`makeidx`' or '`makebib`' respectively, and then the transformation again.

You can't destroy anything: In the worst case one call is waste of time.

3.3.5 Full example with the scripts

Preparation

Let's assume that you want to create a book in tbook format. It is supposed to be called "'`mybook`'". So you create a new subdirectory and enter it with

```
mkdir mybook
cd mybook
```

Now you must prepare the new directory for your project. Thus you enter

```
tbprepare mybook
```

Now you can list all files in the directory with `dir` (Windows) or `ls -l` (Unix). You can see that not only the raw "make" scripts have been created, but also a tiny file '`mybook.xml`' that you can use as a beginning for your document.

Now write the document with your favourite XML or text editor. You may also rename your XML file (but leave the file extension).

Conversion to Postscript and PDF

Eventually you want to create something actually usable from your document. No problem, the command

```
tbtolatex -t mybook "two-column=true"
```

converts '`mybook.xml`' to '`mybook.tex`'. This '`two-column=true`' means that you want to have it typeset with two columns, see Section 4.1 [XSLT parameters], page 17. The '`-t`' means that tbook should assumes that you also (later) want to convert it to HTML 4.

But first, we want to assure that all graphics used in the document are in the correct format. You have already copied all vector images in EPS format, and all bitmaps in JPEG format into the document directory, as explained in Section 3.4 [Original graphics files], page 13. Now call

```
makeepss
```

which converts all JPEGs to EPS. This makes it possible to create with

```
latex mybook
dvips mybook
```

a Postscript version of your book called '`mybook.ps`'. It may be necessary to call '`latex`' multiple times in order to get proper cross-referencing and table of contents.

The command

```
makepdfs
```

creates PDF versions of all EPS'es. Now you are ready to call

```
pdflatex mybook
```

in order to have your book in PDF format ('`mybook.pdf`'). The first pdflatex run may show error messages because it still finds old "ordinary LaTeX" help files but you can ignore that.

Conversion to HTML

Now enter

```
makewebs
```

which creates web-ready bitmaps of all your graphics. With

```
tbtohtml -t mybook
```

you create '`mybook.html`'. You can already open that in your web browser. But it's more convenient to say

```
makepage
```

which creates a new subdirectory '`public-mybook`' and copies all web relevant stuff into that directory. Then you can zip everything in that directory and move it to your web provider. That's it.

Further stuff

If you have equations in your book, you want to have the bitmap equations created in order to see them on an HTML 4 page (remember that MathML is only possible with XHTML). You do this with

```
makeeqns
```

This usually takes a long time. It's *very* much faster if you use the '`preview.sty`' LaTeX package. See Section 4.1 [XSLT parameters], page 17, for how to achieve this.

If you have a bibliography and an index with your document, you have to call

```
makebib
makeidx
```

to create them or, if they are already there, to keep them up-to-date. They create bibliography and index for the *last* format to which you have converted. After that, you have to convert to HTML again or call LaTeX again, respectively, to actually see the changes.

3.4 Your original graphics files

Next to your **t**book XML document, this directory must contain all graphics that you want to include. **t**book knows four image types:

`bitmap`	must be a JPEG, its embedded resolution information is used to determine its real dimensions.
`vector`	must be an EPS, its bounding box determines its real dimensions (of course).
`overlay`	is a JPEG, with an EPS on top of it. The EPS may contain labels etc. Real dimensions as with bitmap/vector. Attention: *You* must assure that both components have the same size. The EPS has the same name as the JPEG, but an '`l`' appended ("labels").
`diagram`	is a LaTeX fragment (e. g. LaTeX picture output by Gnuplot) with the file extension '`.pic`'.

Please note that **t**book may get confused if the same graphics is available in different formats in your texmf searchpath. For example, if you include a JPEG that is also available as a PNG in another directory (but where TeX looks for files), then pdfLaTeX will include that PNG which is surely not what you want.

3.5 Viewing your document in a browser

Unfortunately there still is no perfect way to publish demanding documents in the World Wide Web, especially when they contain math. Therefore **t**book offers a great variety of variants and combinations of them.

You must decide what is best for you. If you are lost, just omit all parameters, because this is probably good enough. However, that needn't be the best option, and the situation will change in the future.

3.5.1 Possible HTML variants with tbook

tbook implements three levels of HTML support at the moment:

1. XHTML. This is the default.
2. HTML 4. This is activated with the '`-t`' option:

   ```
   tbtohtml -t mybook
   ```
3. Quirky HTML. This is activated with the '`-t`' option *and* the XSLT parameter '`css-mode`' set to '`very-careful`':

   ```
   tbtohtml -t mybook css-mode=very-careful
   ```

If you create HTML 4 files with the argument '`-t`', you will be able to savour your text in all modern browsers. If you use that "quirky mode", even NS 4 can be made happy.

3.5.2 XHTML pages

Without the '`-t`' option, **t**book creates XHTML 1.1+MathML files. XHTML is the upcoming standard in the Web. But in order to read XHTML properly with a web browser, two conditions must be met:

1. The browser must be aware of newest web standards, namely XHTML 1.1, MathML 2 and CSS Level 2. At the moment, I know only two browsers already doing so very well: Mozilla 1.0[1] and Netscape 7.

   But older browsers (I tested Netscape 6.2 and Internet Explorer 6) display it quite well (of course not the MathML parts).
2. The web server must serve the page as an xhtml or an xml file, otherwise at least Mozilla doesn't display the MathML components (and it does so for good reason). As far as I understand it, the HTTP `Content-Type` must be set to `text/xml`[2].

XHTML pages that are served as ordinary HTML *and* that don't contain MathML may be displayed quite appealing.

Hickson's Trick: If you insist on proper MathML rendering, an interesting trick is to serve the same XHTML file as HTML for Internet Explorer and as XHTML for all others. Then both big browsers can view the MathML parts: Netscape/Mozilla via intrinsic features, and the IE via the free MathPlayer[3] plugin. For this trick, have a look at `http://software.hixie.ch/utilities/cgi/xhtml-for-ie/`.

3.5.3 The Universal Math Stylesheet

As an alternative to [Hickson's trick], page 14, you can use the Universal Math Stylesheet (UMSS) with tbook. It has the advantage that you don't have to be able to install CGI scripts on your web server, and it is more universal. The disadvantages are that off-line viewing is a little bit inconvenient, and loading of the pages is slower.

See `http://www.w3.org/Math/XSL/` for comprehensive information about the UMSS. There you can also download the four UMSS files. However for tbook, you only need '`pmathml.xsl`' and '`pmathmlcss.xsl`'. Copy both files in the same directory where the XHTML file is. This is also necessary for each copy of your document on a web server.

Then call tbook's XHTML conversion program '`tbtohtml`' with the XSLT parameter '`umss-rendering`' set to '`on`'. That's it.

Unfortunately Mozilla and Netscape use not-very-fast internal XSLT processors. Therefore a test document on my computer took 1.5 seconds to be displayed without the UMSS, and 4.2 s with it.

Other allowed values for '`umss-rendering`' can be found at `http://www.w3.org/Math/XSL/` in the section "Specifying preferences". They are used as values for the `pref:renderer` attribute.

3.6 Jade/nsgmls issues

You can validate your tbook files with nsgmls, an SGML validator that can be called by Emacs for you. It will show error messages, because it bases on an XML declaration file that is erroneous. Try to find this file (on my system it's called '`xml.dcl`', but '`xml.decl`' is also possible) and change the line

```
128    32  UNUSED
```

to

[1] For free download at `http://www.mozilla.org`

[2] actually to `text/xhtml+xml`, but only few applications support this so far

[3] `http://www.dessci.com/webmath/mathplayer`

```
      128     32  128
```

You may also simply ignore those error messages.

But this modification will also help Jade which stands behind the '`docbook2??`' translators. By the way, Jade may complain about namespace attributes, but this too is false alert.

3.7 Other XSLT processors: Saxon 7, Xalan, . . .

At the moment, **t**book is still an XSLT 1.0 application. Eventually it will be 2.0, though. Then, but only then it will support Saxon 7. Unfortunately, I see no possibility to support both Saxon 6.5.*x* and Saxon 7.*x* at the same time.

I don't support Xalan. I've never tried that, maybe it runs through without error messages, but Xalan is not able to create shell scripts, which would make things hopelessly awkward.

All of this is nothing for me to worry about, because Saxon 6.5.*x* is just so easy to install, even parallely with Saxon 7.

As soon as XSLT 2.0 is a finished specification, **t**book will support it. Then you can use **t**book with *all* XSLT processors. Well, all that support XSLT 2.0. But Xalan and Saxon will be surely among them.

4 Configurability

4.1 XSLT parameters

`tbtolatex`, `tbtohtml` and `tbtodocbk` can pass parameters to the XSLT processor Saxon in the form '`<parametername>=<value>`', so e. g.

```
tbtolatex test param1=value1 param2=value2 ...
```

For a list of the available parameters, have a look at tables Section 4.1 [XSLT parameters], page 17 and Section 4.1.1 [Further XSLT parameters], page 18.

For Windows, you have to enclose every parameter pair with `"..."`:

```
tbtolatex test "param1=value1" "param2=value2" ...
```

If you use XSLT parameters, make sure that all transformation scripts get the same set of parameters and values.

For your convenience, you may change these transformation scripts on your local system, but only in order to change the default values of the XSLT parameters. So you can adjust it to your personal taste without being forced to type it every single time. But never distribute such modified files under the original name!

First, the more important XSLT parameters:

'`adr-filename`'
: (Default: '`adrbook.xml`'.) Filename of the address book file.

'`anti-aliasing`'
: (Default: '`true`'.) Default is '`false`' for OS/2. With '`true`', graphics for HTML output are smoothed. Needs the p?m* programs for the PNM image format.

'`assume-os`'
: (Default: '`linux`'.) Assumed operating system. Possible values are '`linux`', '`osx`', '`windows`', and '`os2`'.[1]

'`bib-filename`'
: (Default: '`biblio.xml`'.) Filename of the BibTeXML file.

'`create-image-comments`'
: (Default: current value of `anti-aliasing`.) If '`true`', in all HTML images copyright notices are inserted. In PNGs even image description data. Needs active smoothing, because the p?m* tools are used.

'`css-file`'
: (Default: ''.) Filename of a CSS file (XML format).

'`desperate-measures`'
: (Default: '`false`'.) If '`true`', "desperate measures" are applied, see '`tbookdtd.dtx`'.

'`document-fontsize`'
: (Default: '`10pt`'.) Global font size. 10pt, 11pt or 12pt.

[1] Only '`linux`' and '`windows`' are tested, so the rest will surely fail. `;-)` Well, it affects the look of the shell scripts being generated.

‘`html-equations`’
: (Default: ‘`mathml`’.) With ‘`bitmaps`’ all formulae become bitmaps for HTML.

‘`html-resolution`’
: (Default: ‘`100`’.) Dpi resolution for HTML graphics. You can also use it for global HTML image scaling.

‘`include-image-dimensions`’
: (Default: current value of ‘`anti-aliasing`’.) ‘`true`’ or ‘`false`’. With ‘`true`’, all dimensions of included graphics are written into the HTML output for faster display. Only with active smoothing.

‘`jpeg-quality`’
: (Default: ‘`75`’.) JPEG quality of the HTML output in the usual JPEG unit between 0 and 100.

‘`maximal-alt-length`’
: (Default: ‘`50`’.) Maximal number of characters in the ‘`alt`’ attribute pop-up rectangle in HTML, if you are over the image with the mouse.

‘`preview-latex`’
: (Default: ‘`true`’.) ‘`true`’ makes the equation bitmaps generating process *much* faster.[2]

‘`secnumdepth`’
: (Default: article: ‘`2`’; book: ‘`3`’.) Deepest section level with numbering. -1: part, 0: chapter, 1: section, 2: subsection, etc.

‘`split-level`’
: (Default: ‘`none`’.) If ‘`chapter`’, the HTML file is divided into file chunks for every chapter.

‘`sty-file`’
: (Default: ‘`tbook-pl`’.) Filename of a LaTeX ‘`sty`’ file, ‘`(none)`’ for don’t use any.

‘`tocdepth`’
: (Default: value of ‘`secnumdepth`’.) Deepest section level that gets into the table of contents. ‘`-2`’ switches the TOC off.

‘`transparent-pngs`’
: (Default: ‘`false`’.) Should all white in PNGs become transparent? (Also applies to equation bitmaps.)

‘`two-columns`’
: (Default: ‘`false`’.) If ‘`true`’, the document is set with two columns.

4.1.1 Further XSLT parameters

The following parameters are not so important, but nobody knows. Many of them must be considered internal and shouldn’t be changed by the user.

‘`cat`’ (Default: ‘`cat`’.) Shell command for sending a file to standard output.

[2] Needs ‘preview’ package (`http://preview-latex.sf.net`) and more disk space.

‘`cp`’ (Default: ‘`cp -f`’.) Shell command for file copying.

‘`css-mode`’
: (Default: ‘`careful`’.) If ‘`standard`’, non-simplified CSS are created. On the other hand, ‘`very-careful`’ makes very much simplified CSS for ancient browsers like Netscape 4, but unfortunately Konqueror, too.

‘`debug`’ (Default: ‘`false`’.) If ‘`true`’, you see all LaTeX messages.

‘`docbook-equations`’
: (Default: ‘`text`’.) If ‘`mathml`’, DocBook XML+MathML is created.

‘`dvips-offset`’
: (Default: ‘`0pt,0pt`’.) Dvips offset for the gallery processes.

‘`eqn-gallery-filename`’
: (Default: ‘`<dn>-eqn-gallery`’.) File name of the formulae gallery LaTeX file.

‘`equation-resolution`’
: (Default: value of ‘`html-resolution`’.) Dpi resolution of the bitmap equations.

‘`file-extension`’
: (Default: ‘’.) File extension of generated shell scripts, including a possible leading dot.

‘`gallery-filename`’
: (Default: ‘`<dn>-gallery`’.) File name of the image gallery LaTeX file.

‘`given-document-name`’
: (Default: ‘`tbook-temp-file`’.) Assumed filename of the document, if the correct one can’t be determined (e. g. not Saxon used).

‘`graphics-fontsize`’
: (Default: ‘`default`’.) Font size assumed for all graphics. Default means “same of document”. ‘`10pt`’, ‘`11pt`’ or ‘`12pt`’. See “`basefontsize`” in ‘`tbookdtd.dtx`’.

‘`gs`’ (Default: ‘`gs`’.) How to invoke Ghostscript.

‘`html-extension`’
: (Default: ‘`.xhtml`’.) Assumed file extension for the HTML file.

‘`include-originals`’
: (Default: ‘`true`’.) If ‘`true`’, you can click on bitmaps in HTML and get a view of the original JPEG.

‘`index-at`’
: (Default: ‘`@`’.) Symbol before the sorting key. Also xindy.

‘`index-input-filename`’
: (Default: ‘`<dn>-ind.xml`’.) File name for xindy output (which is XSLT *input.*)

‘`index-markup-separator`’
: (Default: ‘`|`’.) Symbol that introduces markup info in xindy output,

‘`index-output-filename`’
: (Default: ‘`<dn>-idx.xml`’.) File name for xindy input (which is XSLT *output*).

'`index-separator`'
(Default: '`!`'.) Symbol to separate xi̊ndy entry from subentry.

'`latex-quiet-option`'
(Default: '`-interaction=batchmode`'.) How to keep TEX calm.

'`lb`' (Default: ' `\&#10;` '.) Line break within a shell script.

'`mathplayer-pi-location`'
(Default: '`head`'.) Where to put the `<object>` element and the processing instruction for the MathPlayer plugin. The default value makes tbook include it in the head of the XHTML file. For making Konqueror (that has a bug regarding this) happy, use '`body`' as the value for this parameter. The value '`none`' prevents any inclusion of MathPlayer specific code.

'`mkdir`' (Default: '`mkdir -p`'.) Shell command for creating a sub directory.

'`mv`' (Default: '`mv -f`'.) Shell command for file renaming.

'`rem`' (Default: '`# `'.) Command that introduces comments in shell scripts.

'`rm`' (Default: '`rm -f`'.) Default is '`del`' for Windows & OS/2. File deletion command.[3]

'`shift-equations`'
(Default: '`true`', but '`false`' if '`css-mode`' is set to '`very-careful`'.) Should bitmap equations have a corrected baseline? ('`false`' necessary for Netscape 4.77).

'`two-side`'
(Default: article: '`false`'; book: '`true`'.) If '`true`', duplex printing is assumed (e. g., all chapters start on odd pages).

4.2 CSS style sheets

With the XSLT parameter '`css-file`' you can give the file name of a special XML file with the following example contents:

```
<style xmlns="http://www.w3.org/1998/Style/CSS2">
  h1 { color: red }
</style>
```

This would print all first-level headings red. The `<style>` tags must look exactly like this, other XML tags are not allowed. Between them you may insert arbitrary CSS ("Cascading Style Sheets") commands. They are loaded *last* and thus have priority. You can change the final layout of your (X)HTML document in many ways. *Other output formats are not affected.*

Let's have a look at another, more complete example. You want to change all colours in your HTML file. This black on white is boring and too bright for you. Therefore you create a file called '`mybook-css.xml`' with the following contents:

[3] If you don't want that at all, set it e. g. to '`#`'.

```
<style xmlns="http://www.w3.org/1998/Style/CSS2">
    body { background-image: url("blue-tile.png");
           background-color: navy;
           color: white }
    hr.footnoterule { color: white }
    div.title { font-size: x-large; color: yellow }
    div.title-article { font-size: x-large }
    div.partheadline { font-size: x-large }
</style>
```

This sets a navy blue background, and the PNG bitmap '`blue-tile.png`' as the background pattern. (You are responsible that this PNG file exists.) The text colour is now white. Additionally, all parts of the output that used to be '`xx-large`' are now only '`x-large`' because Internet Explorer tends to print them too big. This affects the title (both book and article) and the parts headings.

For more info about the CSS please consult the W3C specification page, and you will find nice books about them in your favourite library/bookstore.

The default CSS commands for tbook documents can be found in '`tbookxsl.pdf`' which is produced with

```
make tbookxsl.pdf
```

in section 2.2.5, "CSS style sheets". But even easier is to have a look at the beginning of an HTML file that tbook produces where you find the whole tbook CSS code. And then you can overwrite the things that you don't like with your own code.

4.3 LaTeX `cfg` file

Is in the LaTeX input path (in particular in the current directory) a file called '`tbook.cfg`', it is included immediately before `\begin{document}`. Typically you will redefine here the macro `\MakeTitlePage`, but you can also change the document font for example:

```
\ifpdflatex\else\ifgallery\else
  \psfonts
\makeatletter

\RequirePackage{geometry}
\@ifpackagelater{geometry}{2002/07/08}{%
  \def\opt{includehead,}}{\def\opt{}}
\expandafter\geometry\expandafter{\opt
               nofoot,a4paper,
               lmargin=2.7cm,rmargin=3cm,     % left and right margin
               tmargin=1.8cm,bmargin=3.9cm,   % top and bottom margin
               headheight=0.5cm,
               headsep=0.5cm,twosideshift=0cm}
\makeatother
\fi\fi
```

The first three lines activate Times font, the rest sets the page margins (for odd pages). Both is only done if we are not producing a PDF (`\ifpdflatex\else`) *and* if we are nor producing a so-called gallery (`\ifgallery\else`). Adjust the values for your needs. You can also use another unit, like “in”.

With such a `cfg` file you can do very similar things as with a LaTeX `sty` file, see Section 4.4 [LaTeX sty file], page 22. But the `cfg` file mechanism is included in tbook to do simple things in a quick and simple way.

4.4 LaTeX sty file

With the XSLT parameter `sty-file` you can include an arbitrary LaTeX package instead of the default ‘`tbook-pl.sty`’. In this file you may change the default behaviour in a cleaner way than it is possible with a `cfg` file. But you have to pay attention to the following:

- If you change page layout (margins), you may only do so if it’s no a *gallery*:

  ```
  \ifgallery\else
  \RequirePackage[...]{geometry}
  \fi
  ```

  See Section 4.3 [LaTeX cfg file], page 21, for a complete example.
- Notice that at the beginning of the document, the command `\pagestyle{fancy}` is included. This means that you should change the style of headers and footers using the `fancyhdr` package. You can assume that the `fancyhdr` package is already loaded.
- You may also change the creating of the title page in `\MakeTitlePage` in your `sty` file. The meaning of the parameters and a template for it can be found in ‘`tbook.sty`’.

You can give the name of the desired LaTeX package in the `class` attribute of your top-level element. So if you say

```
<book class="mystyle.sty">
  ...
```

then ‘`mystyle.sty`’ is included, and *not* the default ‘`tbook-pl.sty`’. The file extension `.sty` in the `class` attribute is mandatory, otherwise it’s ignored in the LaTeX output. However, an explicit `sty-file` parameter on the command line has the highest priority.

4.5 Driver files

Finally, you can overwrite existing templates or add new ones in the so called driver files:

‘`tbtoltx.xsl`’	LaTeX output
‘`tbtohtml.xsl`’	XHTML output
‘`tbtohtml4.xsl`’	HTML 4 output
‘`tbtodb.xsl`’	DocBook XML output

These files are almost empty, actually they only import the really big ones ‘`tblatex.xsl`’, ‘`tbhtml.xsl`’ and ‘`tbdocbk.xsl`’, which are files that you shouldn’t modify.

In DocBook, such files are called “customisation layers”.

Modifications of these files are especially useful for own letter designs, because the default is for a certain Bugs Bunny. For that, redefine the `"letter"` template and – for HTML output – the `"closing"` template.

5 tbook's "almost LaTEX" formula syntax

5.1 MathML v. simplified LaTEX

You can use directly MathML's presentation and contents markup in tbook files. For this, insert `<math>` elements without worrying about namespaces. But except for not too complicated equations this is quite longish: Einstein's famous equation of equivalence of mass and energy looks like

```
<math>
  <mi>E</mi>
  <mo>=</mo>
  <mi>m</mi>
  <msup>
    <mi>c</mi>
    <mn>2</mn>
  </msup>
</math>
```

in MathML. I hope you agree with me that there is room for improvement. Therefore tbook's formula elements `<m>` (inline equation), `<dm>` (displayed or block equation), `<ch>` (chemical formula), and `<unit>` (physical quantity) use a so-called simplified LaTeX syntax.

You may use roots, fractions, standard functions like "sin", stretchable braces, sub- and superscripts and accents. You may nest these structures so deep until your XSLT processor complains. And you can use these elements *inside* MathML to get the best of both worlds. (However HTML output only contains valid MathML.)

Roots work with `\sqrt`, fractions with `\frac`, just as in LaTeX. Human text is included via `\text` that is known from AMSTEX. Standard functions are typed without a '\'. Stretchable braces are all braces immediately within a `{...}` grouping. Sub- and superscripts as in LaTeX, but always a possible subscript *before* the superscript. Accents are just written *immediately* before the accented variable or group, they're made wide accents if necessary. If you make a space between accent and anything that follows, the accent is treated as an operator. (So, a `\vec` becomes a `\to`.)

Here an example:

```
<m>^{1-x_{\text{eff}}} ≠ {( ∫_0^∞ sin(~x)
                              \frac{\sqrt[3]{1/e}}β dx )}
                         ≠ lim_{x → ∞}\frac1x</m>
```

(You can see the special characters only in a Unicode-able editor of course. For other editors you have to use the named entities like `&beta;`.) In printed output this will look like this:

$$\widehat{1-x_{\text{eff}}} \neq \left(\int_0^\infty \sin(\tilde{x})\frac{\sqrt[3]{1/e}}{\beta}dx\right) \neq \lim_{x\to\infty}\frac{1}{x}$$

For HTML output, the responsible stylesheet produces:

```
<math><mover accent="true"><mrow><mn>1</mn><mo>-</mo>
<msub><mi>x</mi><mrow><mtext>eff</mtext></mrow></msub></mrow>
```

```
<mo>&Hat;</mo></mover><mo>&ne;</mo><mrow><mo>(</mo>
<munderover><mo>&int;</mo><mn>0</mn><mo>&infin;</mo></munderover>
<mi>sin</mi><mo stretchy="false">(</mo><mover accent="true">
<mi>x</mi><mo>˜</mo></mover><mo stretchy="false">)</mo>
<mfrac><mrow><mroot><mrow><mn>1</mn><mo>/</mo><mi>e</mi></mrow>
<mn>3</mn></mroot></mrow><mi>&beta</mi></mfrac><mi>d</mi><mi>x</mi>
<mo>)</mo></mrow><mo>&ne;</mo><munder><mi>lim</mi><mrow><mi>x</mi>
<mo>&rarr;</mo><mo>&infin;</mo></mrow></munder><mfrac><mn>1</mn>
<mi>x</mi></mfrac></math>
```

Lucky us.

Notice that you can use `<m>`, `<ch>` and `<unit>` *within* MathML constructs. This is useful if you have to include matrices, but it is especially useful for stacked equations, which LaTeX calls "equation arrays". See Section 5.2.1 [Equation arrays], page 26.

5.2 MathML in tbook

We have already seen that MathML should be avoided where possible. You may use it if you want, for more complicated formulae you must use it, unfortunately. This is true for matrices, but it is especially true for equation arrays and equation numbering.

For a detailed description of MathML, see the MathML specification.

5.2.1 Equation arrays

tbook treats a `<math>` element as an equation array, if it consists of only *one* `<mtable>`, with a `groupalign` attribute *or* one or more `<mlabeledtr>` rows. If you set `groupalign="right center left"`, this leads to an `eqnarray` in LaTeX, and where '&' are in LaTeX, you have to use `<maligngroup/>` in MathML. Else the equations are just stacked and not aligned.

But as already mentioned, you can also use `<m>` inside `<math>`, which is very helpful for equation arrays.

If you use these elements within a MathML equation array, you can generate alignment markers (in LaTeX known as '`&`' signs) with '`#`' signs[1]. Put them where they would be in LaTeX. Although MathML requires such a marker at the very beginning of an equation row, this is not true for LaTeX, and not true for tbook.

Here is an example:[2]

```
<math>
  <mtable groupalign="right center left">
    <mtr>
      <mtd id="test"> <m> 1+1 #=# 2 </m> </mtd>
      <mtd> <m> 4 #=# 2 &CenterDot; 2 </m> </mtd>
    </mtr>
  </mtable>
</math>
```

which is the same as LaTeX's

[1] because this is shorter than `&amp;`

[2] Actually both `<mtd>`s should be in *different* `<mtr>`s according to the MathML specification. You may do this, but it is just additional typing, since it doesn't matter for tbook. However the MathML that tbook *produces* is absolutely free of such sloppyness.

```
\begin{eqnarray}
  1+1 &=& 2 \label{Test} \\
  4   &=& 2 \cdot 2 \nonumber \\
\end{eqnarray}
```

and you don't want to see the HTML/MathML output tbook must create for that.

(By the way, being the only child element of an `<mtd>`, `<m>` is implicitly surrounded by an `<mrow>` which is necessary in this context.)

5.2.2 Equation numbers

A tricky point is equation labelling and numbering. tbook supports three ways of giving an equation a label:

1. A `<math>` element has an `id` attribute. For example:

   ```
   <math id="Einstein">
     <mi>E</mi>
     <mo>=</mo>
     <mi>m</mi>
     <msup>
       <mi>c</mi>
       <mn>2</mn>
     </msup>
   </math>
   ```

   or simply

   ```
   <dm id="Einstein">E=mc^2</dm>
   ```

   You can refer to it with

   ```
   <p>The <ref refid="Einstein">equation</ref> is the famous energy
     equivalence.</p>
   ```

2. An `<mtd>` element with an `id` within an equation array:

   ```
   <math>
     <mtable groupalign="right center left">
       <mtr>
         <mtd id="equation1"> <m> 1+1 #=# 2 </m> </mtd>
         <mtd> <m> 4 #=# 2 &CenterDot; 2 </m> </mtd>
       </mtr>
     </mtable>
   </math>
   ```

   which you can refer to by saying

   ```
   <p>If you want to see a pretty useless equation have a look at
     <ref refid="equation1">equation</ref>.</p>
   ```

3. An `<mlabeledtr>` element with an `id` within an equation array, and the contents of the first `<mtd>` element of such an `<mlabeledtr>` row. This version is pretty obscure, and you cannot refer to it with `<ref>` but only with `<mathref>`.

I would recommend you to use only 1. and 2.

5.2.3 Under- and overbraces

Unfortunately there is no standard way to input an overbrace construct in MathML. In LaTeX you can say

```
$\overbrace{(x_1, x_2, \ldots, x_n)}^{\text{$n$ elements}}$
```

in order to get

$\overbrace{(x_1, x_2, \ldots, x_n)}^{\text{$n$ elements}}$

Its **t**book representation is

```
<mover>
  <m>(x_1, x_2, &hellip;, x_n)</m>
  <mover>
    <mo>&OverBrace;</mo>
    <mrow>
      <mi>n</mi>
      <mtext>&nbsp;elements</mtext>
    </mrow>
  </mover>
</mover>
```

The underbrace works accordingly.

6 Human language support

XML is an international file format. Every XML tag can have an `xml:lang` attribute, although the respective DTD must allow them explicitly. **t**book allows them for almost all its elements. The value of this attribute is a *language tag*. **t**book supports the following XML language tags:

`"en"`	English
`"de"`	German
`"fr"`	French
`"it"`	Italian
`"da"`	Danish
`"nl"`	Dutch
`"pl"`	Polish
`"fi"`	Finnish
`"pt"`	Portuguese
`"no"`	Norwegian (Norsk)
`"sv"`	Swedish
`"ru"`	Russian
`"es"`	Spanish
`"ca"`	Catalan
`"en-US"`	American English
`"en-GB"`	British English
`"no-bok"`	Norsk (Bokmal)
`"no-nyn"`	Nynorsk
`"pt-BR"`	Brazilian
`"de-DE"`	FR German
`"de-AT"`	Austrian
`"de-1901"`	German, traditional orthography
`"de-1996"`	German, new orthography
`"de-DE-1901"`	FR German, traditional orthography
`"de-AT-1901"`	Austrian, traditional orthography

`"de-DE-1996"`
: FR German, new orthography

`"de-AT-1996"`
: Austrian, new orthography

The “special” German orthography tags have been registered with IANA. If you need another language, contact the **t**book maintainer.

You can use other language tags, too, but only the above are interpreted by **t**book for LaTeX output. The language is switched on for the whole element. For example,

```
<chapter xml:lang="de">
  <heading>Einleitung</heading>

  <p>In diesem Text werden wir sehen, daß ...</p>

  <p xml:lang="es">Andalucía es, sobre todo, una región
    agrícola. ...</p>
</chapter>
```

embeds a chapter in German in a document that may be written in a wholly different language. The second paragraph however is in Spanish.

Mostly you will use this tag – if at all – only for the top-level element. For example, an article in French will begin with

```
<article xml:lang="fr">
  ...
```

7 The `style` and `class` attributes

Almost every **t**book element can have a `style` and/or a `class` attribute. This is also true for HTML, and in both formats they have the same meaning. Thus their values are passed without any changes to the HTML file, so you can use that for layout changes in the HTML output.

The `class` attribute takes the name of a layout class. This is actively used in a couple of **t**book elements, e. g. in `<theorem>`, but also in the top-level elements, see Section 4.4 [LaTeX sty file], page 22. However you can make much more use of it in the HTML output.

The `style` attribute takes CSS commands, see Section 4.2 [CSS], page 20. Unfortunately it is impossible to preserve this information in DocBook, and only very few CSS commands can be sensibly translated in LaTeX. Just one of them is indeed realised so far, namely text colour.

7.1 Text colour

There is no element that adds colour to a **t**book document. However you can add a `style` attribute to most elements and put CSS commands in it. This attribute is then passed to HTML, but the *colour* information is interpreted for LaTeX output, too. All different ways how to give colour information in CSS are supported. So you may write

```
<p style="color: red">This is in red.</p>
<p style="color: rgb(0,0,100%)">This is in blue.</p>
<p style="color: rgb(255,255,0)">This is in yellow.</p>
<p style="color: #f783ff">This is a shade of blue.</p>
```

and will get the desired result. You can also add other CSS commands, however, the XSLT processor will only scan for the `color:` property. Background colours are not supported. Page colours are not yet supported. (Although both will work in HTML output, of course.)

8 Bibliography

You can use your old BibTeX file, although minor adjustments could be necessary. It is prefered, although not necessary, that if your BibTeX files contain accented characters or something like this, they should be 8-bit files.

tbook's BibTeX styles (by the way, created with '`custom-bib`') will create decent LaTeX output, but for HTML and DocBook, a filter called `bibfix` tries to make BibTeX's almost-XML output real XML. If you've used strange LaTeX constructs in your BibTeX file, you either have to fix the BibTeX file, or you have to extend '`bibfix.l`', a simple flex scanner, to convert them to XML.

Another possible source of trouble in old BibTeX files is the '`edition`' field. Best is to use an English number in letters like `"third"`.

9 Elements reference

9.1 Minimal example

A minimal **t**book file looks like this:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE book
  PUBLIC "-//Torsten Bronger//DTD tbook 1.5.2//EN"
         "http://tbookdtd.sourceforge.net/tbook152.dtd">
<book xml:lang="en-GB">
  <frontmatter>
    <title>A little book</title>
    <author>Torsten Bronger</author>
  </frontmatter>

  <mainmatter>
    <chapter>
      <heading>First Chapter</heading>

      <p>Small is beautiful.</p>
    </chapter>
  </mainmatter>
</book>
```

9.2 Terms and symbols in the element reference

On the following pages, some special expressions are used in the "Possible Contents". First I explain what they mean.

The term "*inline element*" denotes the following elements:

font manipulation: `<em>`, `<visual>`, `<verb>`,
mathematics: `<m>`, `<math>`, `<ch>`,
cross references: `<cite>`, `<pageref>`, `<ref>`, `<vref>`, `<mathref>`,
index: `<ix>`, `<idx>`, `<indexsee>`,
miscellaneous: `<url>`, `<hspace>`, `<unit>`, `<relax>`, `<wrap>`, `<footnote>`, `<graphics>`, `<latex>`.

The term "*block element*" denotes the following elements:

lists: `<description>`, `<enumerate>`, `<itemize>`,
mathematics: `<math>`, `<dm>`, `<ch>`,
quoted material: `<quote>`, `<blockquote>`, `<verbatim>`, `<verse>`,
miscellaneous: `<p>`, `<multipar>`, `<tabular>`, `<latex>`.

The term "*figure/table*" actually denotes the elements `<figure>` and `<table>`, but where they are allowed, the two "big block" elements `<theorem>` and `<proof>` are allowed, too.

Apart from that, in "Possible contents" the typical symbols of regular expression or EBNF are used:

`+`	at least one

`?`	one or none
`*`	arbitrary many, or none
`|`	or (not exclusively)
`(...)`	grouping; another of these symbols immediately after the group refers to the whole group
`,`	sequence; the order is significant

An example: The element `<section>` has the following "Possible contents":

`heading, (block | float)*, subsection*`

This means in human words: The contents of every `<section>` element must start with *one* `<heading>` element. This is followed by an arbitrary number (also zero) of elements belonging to the block or float category, in arbitrary order. *After* them *may* follow `<subsection>` elements, as many as you wish.

An online version of this element reference is available at
`http://tbookdtd.sourceforge.net/dtd/`.

9.3 Top level elements

`<book>` – a book (root element)

Attributes:

`xml:lang`	human language (default: `"en"`)
`id`	unique name (identifier)
`style/class`	style properties and class (e. g. for CSS)

Possible Contents: `frontmatter, mainmatter, backmatter?`

Description: This element embraces all other elements of a *book*. (Like the `<html>` tag does in HTML.)

If a `class` attribute is given, and its contents ends with '`.sty`', the whole thing is interpreted as a LaTeX style file that's included instead of '`tbook-pl.sty`'. However, an explicit style file as an XSLT parameter `sty-file` still has higher precedence. Same with `<article>`.

Example

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE book PUBLIC "-//Torsten Bronger//DTD tbook 1.5.2//EN"
                      "http://tbookdtd.sourceforge.net/tbook152.dtd">
<book xml:lang="en-GB">
  <frontmatter>
    <title>A little book</title>
    <author>Torsten Bronger</author>
  </frontmatter>

  <mainmatter>
```

```
    <chapter>
      <heading>First Chapter</heading>

      <p>Small is beautiful.</p>
    </chapter>
  </mainmatter>
</book>
```

`<frontmatter>` – general info about a book

No attributes.

Possible Contents: `title, author+, subtitle?, date?, keywords?, year?, city?, graphics?, typeset?, legalnotice?`

Description: Everything that is normally mentioned before the table of contents. Only title and one author is a must, the rest if you wish, but the given order is (unfortunately) mandatory. The first mentioned author will also be printed after the copyright.

`<author>` has to be simple, i. e. it mustn't contain any additional information such as institute or email address. If it does, that is ignored. (In contrast to in articles.)

Example

```
<frontmatter>
  <title>Beyond Gravitation</title>
  <author>Wyle E Coyote</author>
  <author>Roadrunner</author>
  <subtitle>An Introduction</subtitle>
  <date>23. July 1968</date>
  <keywords>ACME, levers, umbrellas, earthquake pills,
            rocks</keywords>
  <year>1969</year>
  <city>Toontown</city>
  <graphics kind="bitmap" file="coverimage"/>
  <typeset>The authors.</typeset>
  <legalnotice>All rights reserved.</legalnotice>
</frontmatter>
```

`<mainmatter>` – the text contents

No attributes.

Possible Contents: `(part | chapter)*, appendix?`

Description: All parts and chapters of a *book*.

Example

```
<mainmatter>
  <chapter kind="preface">
    <heading>Preface</heading>
    <p>...</p>
  </chapter>

  <!-- Here will be the table of contents. -->

  <chapter kind="introduction">
    <heading>Introduction</heading>

    <p>...</p>
  </chapter>

  <part id="sec:Preparation">
    <heading>Preparation</heading>

    <chapter id="sec:Samples">
      <heading>Samples</heading>

      <p>...</p>
    </chapter>

    <chapter id="sec:Equipment">
      <heading>Equipment and Techniques</heading>

      <p>...</p>
    </chapter>

  </part>

  <appendix>

    <chapter id="sec:Blueprints">
      <heading>Blueprints</heading>

      <p>...</p>
    </chapter>

  </appendix>
</mainmatter>
```

`<backmatter>` – references and index

No attributes.

Possible Contents: `references?, index?`

Description: Bibliography and index. Both optional, but pay attention to order.

Example

```
<backmatter>
  <references bibfile="mybib"/>
  <index>
    <p>Command names are printed in bold face.</p>
  </index>
</backmatter>
```

`<article>` – an article (root element)

Attributes:

`xml:lang`	human language (default: `"en"`)
`id`	unique name (identifier)
`style/class`	style properties and class (e. g. for CSS)

Possible Contents: `title, author+, date?, keywords?, year?, abstract?, (`*`block element`*` | `*`figure/table`*`)*, section*, references?`

Description: Embraces all elements of an *article*. Title and one author are mandatory. Here – in contrast to book – the `<author>` tag can contain `<newline>` and `<footnote>` elements for e. g. institute or email address.

As for the `class` attribute, see `<book>` above.

Example

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE book PUBLIC "-//Torsten Bronger//DTD tbook 1.5.2//EN"
                      "http://tbookdtd.sourceforge.net/tbook152.dtd">
<article>
  <title>The Grand Unifying Theory</title>
  <author>Thomas Ferguson</author>
  <keywords>life, universe, everything</keywords>

  <abstract>
    <p>Six by nine equals forty-two.</p>
  </abstract>

  <section>
    <heading>Motivation</heading>

    <p>Motivation is simple ...</p>
  </section>
</article>
```

`<letter>` – a letter (root element)

Attributes:

`from`	unique signature of the author
`formal`	`"true"` for “formal letter” or `"false"`
`xml:lang`	human language (default: `"en"`)
`id`	unique name (identifier)
`style/class`	style properties and class (e. g. for CSS)

Possible Contents: `city`, `date`, `to`, `subject`, `opening`, *`block element`*`*`, `closing`

Description: Embraces all elements of a *letter*.

`from` can be e.g. `"Torsten Bronger"`. It should be *your* personal id and should be globally unique, as far as one can guarantee that. Because this attribute is used by the stylesheets to ensure that they are responsible for you (correct letter head etc).[1] Moreover the `owner` attribute in the address book must match it.

If you don't set `formal` explicitly, the default is taken from the address book entry (even if you gave an explicit "to"-address and so overruled the address book entry). If even that fails, `"false"` (private letter) is assumed.

Example

```
<letter from="Torsten Bronger &lt;torsten.bronger@gmx.de&gt;">
  <city>Aachen</city>
  <date>2. September 2002</date>
  <to nickname="Jupp"/>
  <subject>Party!</subject>

  <opening>Hi Jupp,</opening>

  <p>...</p>

  <closing>Till then,</closing>
</letter>
```

9.4 Parts, chapters and sections

`<heading>` – of a chapter etc.

Attributes:

`xml:lang` human language (inherited)

`id` unique name (identifier)

`style/class`
style properties and class (e.g. for CSS)

Possible Contents: `(`*`Text`*` | `*`inline element`*`)*`

Description: All structuring elements (`<part>`, `<chapter>`, `<section>` etc.) begin with this element. It contains of course the title of that section.

Example

See [`<chapter>`], page 41.

[1] In a letter system that is better configurable than the current, this attribute is used to read in the correct parameters such as the letter head.

`<part>` – of a book

Attributes:

`xml:lang`	human language (inherited)
`id`	unique name (identifier)
`style/class`	style properties and class (e. g. for CSS)

Possible Contents: `heading, chapter*`

Description: Embraces a part of a book.

Example

See [`<mainmatter>`], page 37.

`<chapter>` – of a book

Attributes:

`kind`	`"preface"`, `"introduction"`, `"acknowledgements"` or `"colophon"` – marks chapters with special meaning
`xml:lang`	human language (inherited)
`id`	unique name (identifier)
`style/class`	style properties and class (e. g. for CSS)

Possible Contents: `heading, aphorism?, (`*`block element`*` | `*`figure/table`*`)*, section*`

Description: A chapter of a book. It's always included into the TOC (i. e., there is no star'ed version). In `<aphorism>` you can let a nice witty quote by a famous person being printed above the chapter beginning.

`kind` marks special chapters that are not included into the TOC. A `"preface"` chapter is printed *before* the TOC; so far, there is no difference between `"acknowledgements"` and `"colophon"` yet.

Example

```
<chapter kind="introduction">
  <heading>Introduction</heading>

  <p>Before we begin ...</p>

</chapter>

<chapter>
```

```
      <heading>Crystal Growth</heading>

      <p>This can be a tricky thing. ...</p>

      <section>
        <heading>First Steps</heading>

        <p>First we take warm water, ...</p>

      </section>
    </chapter>
```

`<section>`

Attributes:

`xml:lang` human language (inherited)

`id` unique name (identifier)

`style/class`
style properties and class (e. g. for CSS)

Possible Contents: `heading, (`*`block element`*` | `*`figure/table`*`)*, subsection*`

Description: Encloses a section that is included into the table of contents in any case. See [`<chapter>`], page 41.

`<subsection>`

Possible Contents: `heading, (`*`block element`*` | `*`figure/table`*`)*, subsubsection*`

This element is totally analogous to [`<section>`], page 42, it can contain the next lower level of structuring. `<subsection>` wanders in the TOC by default.

`<subsubsection>`

Possible Contents: `heading, (`*`block element`*` | `*`figure/table`*`)*, paragraph*`

This element is totally analogous to [`<section>`], page 42, it can contain the second next lower level of structuring.

`<paragraph>`

Possible Contents: `heading, (`*`block element`*` | `*`figure/table`*`)*, subparagraph*`

This element is totally analogous to [`<section>`], page 42, it can contain a lower level of structuring, below `<subsubsection>`. `<paragraph>` doesn't get in the TOC by default.

`<subparagraph>`

Possible Contents: `heading, (`*`block element`*` | `*`figure/table`*`)*`

This element is by and large analogous to [`<section>`], page 42, but it can't contain a next lower level of structuring, because it's the last one. `<subparagraph>` doesn't get in the TOC by default.

`<appendix>`

Attributes:

`xml:lang`	human language (inherited)
`id`	unique name (identifier)
`style/class`	style properties and class (e. g. for CSS)

Possible Contents: `chapter*`

Description: This element encloses all appendix chapters. They are unusual only as far as their numbering is concerned. So this element is very similar to LaTeX's `appendix` environment.

Example

See [`<mainmatter>`], page 37.

9.5 Paragraphs & friends

`<p>` – paragraph

Attributes:

`skip`	skip before the paragraph: `"small"`, `"med"` or `"big"`
`xml:lang`	human language (inherited)
`id`	unique name (identifier)
`style/class`	style properties and class (e. g. for CSS)

Possible Contents: `(`*`Text`*` | `*`inline element`*` | `*`block element`*`)*`

Description: Embraces *one* paragraph. Empty lines produces a warning and are ignored.

Example

See [`<chapter>`], page 41.

`<multipar>` – sequence of simple paragraphs

Attributes:

`xml:lang` human language (inherited)

`id` unique name (identifier)

`style/class`
style properties and class (e. g. for CSS)

Possible Contents: `(`*`Text`*` | `*`inline element`*`)*`

Description: This one can contain plain text and inline elements. The idea is that every `*` ends a paragraph and begins a new one, so you don't have to type all this `<p>...</p>` stuff every time. Empty lines are ignored, but produce no warning.

Example

```
<multipar>
  The <em>SI</em> developed from the metric system but today both
  terms can be used synonymously. It is worth mentioning that all
  metric countries have a different set of legal units, including some
  non-metric ones. The SI, in most cases, represents the core of the
  metric system, though.*

  The science that deals with measuring quantities and definition of
  units is metrology. Many countries run metrological institutes and
  partly they work as standardising authorities in their respective
  domains.*

  In regular international meetings these institutes decide about
  (re-)definitions of SI units. Since 1983 the seven base units have
  been defined as follows:*

  ...
</multipar>
```

`<newline>` – line break

Attributes:

`vspace` vertical skip

Possible Contents: None.

Description: Inserts a line break. In `vspace` you can give a following skip. It is only allowed at a few places.

`<footnote>`

Attributes:

`xml:lang`	human language (inherited)
`id`	unique name (identifier)
`style/class`	style properties and class (e. g. for CSS)

Possible Contents: `(`*`Text`*` | `*`inline element`*`)*`

Description: Inserts a footnote at the current position. It embraces the footnote text.

Example

```
<p>Zaphod Beeblebrox<footnote>the president of the
   galaxy</footnote> was a friend of him.</p>
```

9.6 Font change

`<em>` – emphasis

No attributes.

Possible Contents: `(`*`Text`*` | `*`inline element`*`)*`

Description: Like `\emph{...}` in LaTeX. It highlights a word or a phase that you want to put emphasis on, usually by setting it in italics.

Example

```
<p>This is a <em>very</em> important point.<p>
```

`<visual>` – visual markup

Attributes:

`markup`	`"nm"`, `"rm"`, `"it"`, `"sc"`, `"bf"`, `"sf"`, `"sl"`, `"tt"`, `"vs"` (required)

Possible Contents: `(`*`Text`*` | `*`inline element`*`)*`

Description: Allows font style/variant change. The attribute `markup` is the same as in LaTeX's `\text??` commands.

`"nm"`	switches back to normal upshape letters. It doesn't change the font *family* that is set by `"rm"`, `"sf"`, and `"tt"`.
`"rm"`	activates Roman (serif) font family. This may be Times or Palatino for example.
`"it"`	switches to the italic variant.
`"sc"`	switches to the small caps variant that is frequently used to typeset names of people.

`"bf"`	switches to the bold face variant.
`"sf"`	activates the sans serif font family. This may be Helvetica or Frutiger for example.
`"sl"`	activates the slanted (oblique) variant. For many fonts this is the same as italics.
`"tt"`	activates the monospaced (typewriter) font family. This may be Courier or cmtt for example.
`"vs"`	activates the versaliae (all-uppercase) font variant. This is intended for acronyms like "BASIC".

Every markup command leaves as much as possible intact from the previous active markup. For example, a `<visual markup="it">` in bold text will switch to bold italics.

Of course, this element can hardly be mapped on DocBook elements. It is mostly realised via DocBook's `<emphasis>`.

Example

```
<p><visual markup="bf">bold</visual>,
   <visual markup="it">italic</visual>,
   <visual markup="sc">Small Caps</visual>,
   <visual markup="sf">sans serif</visual>,
   <visual markup="sl">slanted</visual>,
   <visual markup="tt">typewriter</visual>,
   <visual markup="vs">versaliae</visual>.</p>
```

9.7 Cross references

`<ref>` – simple cross reference

Attributes:

`refid`	ID of the element you want to point to (required)

Possible Contents: `(`*`Text`*` | `*`inline element`*`)*`

Description: This does the same as the LaTeX command `\ref`. The referenced object, that has the ID of `refid`, must be a figure/table, section etc., simply something that can bear a number. The contents of `<ref>` is the textual label that is put immediately before it, e. g.

```
<table id="MainTable">
  ...
</table>

... <ref refid="MainTable">table</ref>
```

This yields something like '`table~2.1`' for LaTeX, or for HTML '`table 2.1`' which is *completely* displayed as the link, and not only the '`2.1`'.

`<ref>`s to equations put automatically parentheses around the number.

`<vref>` – cross reference with page

Attributes:

`refid` id of the element you want to point to (required)

Possible Contents: `(`*`Text`*` | `*`inline element`*`)*`

Description: See `<ref>`, but here we use LaTeX's varioref package which means that the page number is included into the reference. For HTML, there is no difference between `<ref>` and `<vref>`.

`<pageref>` – page reference

Attributes:

`refid` id of the element you want to point to (required)

Possible Contents: `(`*`Text`*` | `*`inline element`*`)*`

Description: Inserts the page number of the object with the ID of `refid`. In HTML a '`[here]`' is inserted (which you can click on), in LaTeX a possibly given contents of the `<pageref>` is inserted directly before the page number. This means that

```
See <pageref refid="GUTformula">page</pageref>.
```

yields '`See page~42`' in LaTeX and '`See [here].`' (clickable `[here]`) in HTML.

`<cite>` – bibliographic reference

Attributes:

`refid` label(s) of bibliographic entries you want to point to (required, if more than one, then separated by spaces)

`kind` `"text"`, `"paren"`, `"imparen"` or `"nocite"`

Possible Contents: `(`*`Text`*` | `*`inline element`*`)*`

Description: The element `<cite>` is used to refer to other published or unpublished material. Before you can use it, you have to build a bibliographic database *outside* tbook. Eventually this will be in an XML format, but till then tbook uses BibTeX for this purpose.

So the first step is to create a BibTeX '`.bib`' file. Please have a look at the BibTeX documentation or at a good LaTeX book for how to do that. The '`bib`' file must be at a place where BibTeX finds it.

Normal citation

The element

```
<cite refid="Williams_Kelley1999"/>
```

inserts a citation reference to the bibliographic entry with the label `"Gruber2001"` at the current location. The `refid` can also be a space separated list of more than one citation:

```
For further reading, see <cite refid="Williams_Kelley1999 Crawford1998
  Nelson1989"/>.
```

This refers to the following BibTeX entries in the ‘`bib`’ file:

```
@MISC{Williams_Kelley1999,
   author = "Thomas Williams and Colin Kelley and Russel Lang and Dave Kotz
                 and John Campbell and Gershon Elber and others",
   title = "Gnuplot",
   month = oct,
   year = 1999,
   url = "http://www.gnuplot.org",
   note = "Das Programm",
}

@MANUAL{Crawford1998,
   author = "Dick Crawford",
   title = "{g}nuplot, an Interactive Plotting Program",
   month = dec,
   year = 1998,
   url = "http://www.gnuplot.org",
}

@BOOK{Nelson1989,
   author = "Ross P. Nelson",
   title = "{Programmierhandbuch $80386$ -- Assemblerprogrammierung mit dem
            Mikroprozessor}",
   publisher = "Vieweg \& Sohn, Braunschweig",
   year = 1989,
}
```

The attribute `kind` determines the kind of insertion. For example,

```
<cite refid="Williams_Kelley1999" kind="text"/>
```

would be printed as

Williams et. al. (1999)

whereas `"paren"` makes

(Williams et. al. 1999)

`"imparen"` (“implicit parentheses”) produces

Williams et. al. 1999

If you don’t give an explicit `kind` attribute, tbook tries to find the best choice according to the context. This means that implicit parentheses are assumed if the `<cite>` element is embraced by parentheses, and `"text"` otherwise.

You know these “kinds” probably from the natbib package, that’s behind all this.

Cite without citing

`"nocite"` includes nothing in the text, just in the references list at the end:

```
    <cite refid="Bathe1986" kind="nocite"/>
```

By the way,

```
<cite refid="-" kind="nocite"/>
```

works like `\nocite{*}` LaTeX, i. e. all entries in your '`bib`' file are included in the references list, explicitely cited or not. (An '`*`' is not allowed in `refid`.)

Optional contents

The contents of the `<cite>` element becomes the optional parameter of LaTeX's `\cite` command. Thus

```
<cite refid="Einstein1921"><em>first</em> chapter</cite>
```

yields:

Einstein (1921, *first* chapter)

`<mathref>` – "odd" reference of a formula

Attributes:

`refid`	ID of the equation you want to reference (required)

Possible Contents: `(`*`Text`*` | `*`inline element`*`)*`

Description: See `<ref>`, but `<mathref>` only works for equations that have been labeled using the first column of an `<mlabeledtr>`. Actually such an equation should also have a real `id`, so you don't need `<mathref>`. It's just to make MathML support a little bit more complete.

(I needed an extra element for this kind of reference because I really wanted to be able to point to such equations, but for validation issues `refid` can't be of type `ID`, but must be `CDATA`.)

9.8 LaTeX-like mathematics

`<m>` – inline equation

Attributes:

`id`	unique name (identifier)
`style/class`	style properties and class (e. g. for CSS)

Possible Contents: *`Text`*

Description: Corresponds to `$...$` in LaTeX, but uses a somewhat different expression syntax, see Chapter 5 [Almost LaTeX formulae], page 25. *Attention:* If it has an `id`, it's becoming a *displayed* Formula, with a number.

You can use `<m>` inside MathML's `<math>`. The same applies to `<ch>` and `<unit>`.

In principle, you can use `<m>` wherever you can use `<mrow>`. However, sometimes it may be necessary to enclose `<m>` and friends with `<mrow>`, because they may expand to multiple MathML elements.

`<dm>` – displayed equation

Attributes:

`id`	unique name (identifier)
`style/class`	style properties and class (e. g. for CSS)

Possible Contents: *`Text`*

Description: Corresponds to `\[...\]` in LaTeX, but uses a somewhat different expression syntax, see Chapter 5 [Almost LaTeX formulae], page 25.

`<ch>` – chemical formula

Attributes:

`display`	`"inline"` or `"block"`
`id`	unique name (identifier)
`style/class`	style properties and class (e. g. for CSS)

Possible Contents: *`Text`*

Description: Corresponds to `<m>` or `<dm>` (according to `display`), but produces chemical formulae with e. g. unslanted chemical element symbols, see Chapter 5 [Almost LaTeX formulae], page 25. e. g.

```
<ch>Al_xGa_{1-x}As</ch>
```

yields "$\mathrm{Al}_x\,\mathrm{Ga}_{1-x}\,\mathrm{As}$", i. e. tiny skips between the elements, upshape elements, but all subscripts are treated as mathematics.

For the explicit namespace see `<m>` above. It makes it possible to be used inside MathML's `<math>`.

9.9 Theorems and proofs

`<theorem>` – (mathematical) theorem etc.

Attributes:

`countlike`
  other theorem's class name, or `"(none)"`, or `"(global)"`

`layout` `"plain"` (default), `"definition"`, or `"remark"`

`xml:lang` human language (inherited)

`id` unique name (identifier)

`style/class`
  style properties and class (e. g. for CSS)

Possible Contents: `heading?, subject?, (`*`Text`*` | `*`inline element`*` | `*`block element`*`)*`

Description: Inserts a (mathematical) theorem, corollary, lemma, definition, or a remark, a note, an exercise, or an example. It is allowed everywhere where you could insert a floating figure.

For this element, the `class` attribute is vital. It is interpreted as the kind of the theorem. e. g., you could say

```
<theorem class="Remark">A short mathematical remark.</theorem>
```

If `class` is omitted a default is used.

The `countlike` contains another `<theorem>` class with which the current one should share numbering. Otherwise, every `<theorem>` class gets its own counting. For example,

```
<theorem class="Corollary"
  countlike="Lemma">A short mathematical corollary.  If
  it's corollary number 4, the next lemma will have
  number 5.</theorem>
```

If you want to suppress counting, set `countlike` to `"(none)"`. If you don't want to have the chapter number included into the theorem number, set `countlike` to `"(global)"`.

Special effects

The name of the theorem is the same as its class. If you want another name, include a `<heading>` element.

Within `<subject>...</subject>`, you can include e. g. a special name for the theorem. This is printed within brackets after the word “Theorem” (or whatever).

Example:

```
     <theorem class="Korollar">
       <heading>Corollary</heading>
       <subject>Streetmentioner's Corollary</subject>
       A short mathematical corollary.</theorem>
```

With the `layout` attribute, you can choose another style. Predefined are the AMSTEX styles `"plain"`, `"definition"`, and `"remark"`. If you want to add another one, you have to give its definition in a LATEX package.

Important: Only the very first occurrence of a certain `<theorem>` class is allowed to have `<heading>`, `countlike`, or `layout`.

`<proof>` – mathematical proof

Attributes:

`xml:lang`	human language (inherited)
`id`	unique name (identifier)
`style/class`	style properties and class (e. g. for CSS)

Possible Contents: `heading?, (`*`Text`* `|` *`inline element`* `|` *`block element`*`)*>`

Description: Inserts a mathematical proof. It is allowed everywhere where you could insert a floating figure.

If `<heading>` is given, it is used instead of the default word “Proof” at the beginning.

Example

```
<proof>
  Indirect: ...

  <p>(Some magic happens.)</p>

  <p>This contradicts
    <ref refid="eqn:variance">equation</ref>.</p>
</proof>
```

This yields:

Proof. Indirect: ...

(Some magic happens.)

This contradicts equation (4).

9.10 Miscellaneous

`<url>` – external reference

Attributes:

`name`	the URL (required)

Possible Contents: `(`*`Text | inline element`*`)*`

Description: The contents of this element is hyperlinked with `name`. For printed output, the URL is printed in parentheses after the contents.

If the element is empty, the `name` in printed with typewriter style and is linked with itself.

Example

```
<url name="http://www.w3c.org">W3C</url>
```

This yields

W3C

whereas

```
<url name="http://www.w3c.org"/>
```

leads to

`http://www.w3c.org`

<hspace> – horizonal skip

Attributes:

`dim`	width (required)

Possible Contents: None.

Description: Makes a horizontal skip, as the LaTeX macro of the same name.

Example

```
Hello<hspace dim="1em"/>world
```

yields

Hello world

<relax>

No attributes.

Possible Contents: None.

Description: Does nothing. Had a meaning between `<cite>`s in former times, when `<cite>` was defined differently. But I didn't want to abandon it. Who knows what it still can be good for. And I dislike input languages without a `<relax>`.

`<unit>` – physical quantity

No attributes.

Possible Contents: *`Text`*

Description: Inserts a physical quantity, see Chapter 5 [Almost LaTeX formulae], page 25.

```
<unit>3 m</unit>
```

yields in LaTeX '`$3$\,m`'. So it guarantees a neat skip between number and unit, and for configurations with different fonts for number in- and outside mathematics it chooses the correct one. Further advantage: Things like

```
The gravitational constant is
<unit>6.672&middot;10^{-11} m^3 kg^{-1} s^2</unit>.
```

(notice the spaces!) yields

The gravitational constant is $6.672 \cdot 10^{-11}\,\mathrm{m}^3\,\mathrm{kg}^{-1}\,\mathrm{s}^2$.

So, units in upshape with small skips inbetween. You must assure that the *first* space is between the number and the unit, or alternatively you must put a '`~`' between number and unit.

For the explicit namespace see `<m>` above. It makes it possible to be used inside MathML's `<math>`.

`<latex>` – LaTeX code

Attributes:

`code`	LaTeX code (required)
`desperate`	`"true"` or `"false"` (default) – element for last phase of production?

Possible Contents: Arbitrary.

Description: This is some sort of emergency command: It inserts `code` if we're producing LaTeX output, and interprets the element's contents else. It's totally ignored if `desperate` is `"true"`.

As an example I mention here the definition of the tbook predefined entity '`&LaTeX;`':

```
<!ENTITY LaTeX "<latex code='\LaTeX{}'>LaTeX</latex>">
```

Then `&LaTeX;` prints a neat LaTeX logo in all outputs.[2]

The contents of this element is tbook code, no HTML code!

[2] Well, as good as the respective output can do it.

The `desperate` attribute

The idea behind `desperate` is that the transformation tbook ⇒ LaTeX can be forced to interpret even `<latex>`'es with `desperate="true"` if a special XSLT parameter called '`desperate-measures`' is set to '`true`', see Section 4.1 [XSLT parameters], page 17. Have a look at an example:

```
<latex code="\newpage" desperate="true"/>
```

This is usually done in the very last phase of production, maybe for having better page breaks like in this example. Normally it is totally ignored, but if you call the LaTeX transformation with

```
tbtolatex mybook desperate-measures=true
```

the (hopefully) optimising page breaks are applied.

`<wrap>` – inline wrapper

Attributes:

`xml:lang`	human language (inherited)
`id`	unique name (identifier)
`style/class`	style properties and class (e. g. for CSS)

Possible Contents: `(`*`Text`*` | `*`inline element`*`)*`

Description: This doesn't format, it encloses inline elements that then get an `id` or a language via `xml:lang`. It can be useful empty. For example

```
<wrap id="NicePosition"/>
```

is equal to LaTeX's `\label{NicePosition}`. So you can refer to a special position in your text with `<pageref>`.

9.11 Quoted and verbatim material

`<quote>` – inline quotation

Attributes:

`xml:lang`	human language (inherited)
`id`	unique name (identifier)
`style/class`	style properties and class (e. g. for CSS)

Possible Contents: `(`*`Text`*` | `*`inline element`*`)*`

Description: Inserts an inline quotation, and adds the quotation marks according to the current language.

Example

```
<p>An old German country saying:
<quote xml:lang="de">Sind die Hühner platt wie Teller, war
  der Traktor wieder schneller.</quote></p>
```

Yields:

An old German country saying: „Sind die Hühner platt wie Teller, war der Traktor wieder schneller.“

`<blockquote>` – displayed quotation

Attributes:

`xml:lang`	human language (inherited)
`id`	unique name (identifier)
`style/class`	style properties and class (e. g. for CSS)

Possible Contents: *`block element*`*

Description: Inserts a displayed quotation.

Example

```
<p>An old German country saying:
<blockquote xml:lang="de"><p>Sind die Hühner platt wie Teller, war
  der Traktor wieder schneller.</p></blockquote></p>
```

Yields:

An old German country saying:

> Sind die Hühner platt wie Teller, war der Traktor wieder schneller.

`<verb>` – preformatted inline material

No attributes.

Possible Contents: *`Text`*

Description: Basically the same as LaTeX's `\verb`. It typesets its contents in typewriter style and avoid line breaks.

Example

```
<p>Every Pascal program starts with
  the keyword ‘<verb>program</verb>’.</p>
```

yields

> Every Pascal program starts with the keyword ‘`program`’.

`<verbatim>` – preformatted displayed material

Attributes:

`xml:lang`	human language (inherited)
`id`	unique name (identifier)
`style/class`	style properties and class (e. g. for CSS)

Possible Contents: `(`*`Text`*` | em | visual | ix | idx | indexsee)*`

Description: Corresponds to LaTeX's `verbatim` environment. In this context XML's `<![CDATA[...]]>` is sometimes useful. Please note that – in contrast to LaTeX – some formatting elements are allowed. So you may print things in bold face, for example.

At least for LaTeX output you can activate syntax highlighting via the `class` attribute:

```
<verbatim class="C++">
int main() {
  cout << "Hello, world!\n";
  return 0;
}
</verbatim>
```

This may be printed as

```
int main() {
  cout ≪ "Hello, world!\n";
  return 0;
}
```

The `class` attribute can take all values that are understood by the listings package of LaTeX.

`<verse>` – lyrics

Attributes:

`xml:lang`	human language (inherited)
`id`	unique name (identifier)
`style/class`	style properties and class (e. g. for CSS)

Possible Contents: `(`*`Text`*` | `*`inline element`*`)*`

Description: Formats its contents in a way that line breaks are conserved, which is significant for e. g. lyrics.

Example

```
<verse xml:lang="de">Hat der alte Hexenmeister
  Sich doch einmal wegbegeben.
  Und nun sollen seine Geister
  Auch nach meinem Willen leben.
  Seine Wort und Werke
  Merkt ich und den Brauch.
  Und mit Geistesstärke
  Tu ich Wunder auch.
</verse>
```

This yields:

Hat der alte Hexenmeister
Sich doch einmal wegbegeben.
Und nun sollen seine Geister
Auch nach meinem Willen leben.
Seine Wort und Werke
Merkt ich und den Brauch.
Und mit Geistesstärke
Tu ich Wunder auch.

`<aphorism>` – an epigraph

Attributes:

`xml:lang`	human language (inherited)
`id`	unique name (identifier)
`style/class`	style properties and class (e. g. for CSS)

Possible Contents: `(`*`Text`*` | `*`inline element`*`)*, caption?`

Description: Embraces a little witty quote for the beginning of a chapter. The element must be immediately after the `<heading>...</heading>` of that chapter. `<caption>` contains the origin, typically the name of a more or less famous person. There must be up to *one* `<caption>` and it must come *last* within `<aphorism>`.

Example

```
<aphorism>&ldquo;Forty-two,&rdquo; said Deep Thought with
  infinite majesty and calm.
  <caption>Douglas Adams,
    <em>The Hitchhiker's Guide to the Galaxy</em></caption>
</aphorism>
```

This yields:

"Forty-two," said Deep Thought with infinite majesty and calm.
—Douglas Adams, *The Hitchhiker's Guide to the Galaxy*

9.12 Lists

<itemize>

Attributes:

`xml:lang`	human language (inherited)
`id`	unique name (identifier)
`style/class`	style properties and class (e. g. for CSS)

Possible Contents: `item*`

Description: A not numbered list of `<item>`s.

Example

```
<p>The most important problems are:
  <itemize>
    <item>rough edges</item>
    <item>small etch holes</item>
    <item>wriggling notch</item>
  </itemize>
</p>
```

This yields:

The most important problems are:

- rough edges
- small etch holes
- wriggling notch

<enumerate>

Attributes:

`xml:lang`	human language (inherited)
`id`	unique name (identifier)
`style/class`	style properties and class (e. g. for CSS)

Possible Contents: `item*`

Description: A numbered list of `<item>`s.

Example

```
<p>The three laws of Captain Ed Murphy:</p>

<enumerate>
  <item>Nothing is as easy as it looks.</item>
  <item>Everything takes longer than you think.</item>
  <item>If anything can go wrong, it will.</item>
</enumerate>
```

This yields:

The three laws of Captain Ed Murphy:

1. Nothing is as easy as it looks.
2. Everything takes longer than you think.
3. If anything can go wrong, it will.

`<description>` – glossary like list

Attributes:

`xml:lang`	human language (inherited)
`id`	unique name (identifier)
`style/class`	style properties and class (e. g. for CSS)

Possible Contents: `(term, item)*`

Description: A glossary like list of `<term>`–`<item>` pairs.

Example

```
<description>
  <term>Red</term>
    <item>Pretty colour at the low energy end of the visible
      electromagnetic spectrum.</item>

  <term>Nine</term>
    <item>Odd number exactly between eight and ten.</item>

  <term>Rigel</term>
    <item>Star in the constellation of Orion.  A blue giant
      of spectral class B.</item>
</description>
```

This yields:

Red	Pretty colour at the low energy end of the visible electromagnetic spectrum.
Nine	Odd number exactly between eight and ten.

Rigel	Star in the constellation of Orion. A blue giant of spectral class B.

`<item>` – list item

Attributes:

`xml:lang`	human language (inherited)
`id`	unique name (identifier)
`style/class`	style properties and class (e. g. for CSS)

Possible Contents: `(`*`Text`*` | `*`inline element`*` | `*`block element`*`)*`

Description: May be used within [`<itemize>`], page 59, [`<enumerate>`], page 59, and [`<description>`], page 60.

`<term>` – description term

Attributes:

`xml:lang`	human language (inherited)
`id`	unique name (identifier)
`style/class`	style properties and class (e. g. for CSS)

Possible Contents: `(`*`Text`*` | `*`inline element`*`)*`

Description: See [`<description>`], page 60.

9.13 "Floats" and their contents

`<figure>` – floating figure

Attributes:

`xml:lang`	human language (inherited)
`id`	unique name (identifier)
`style/class`	style properties and class (e. g. for CSS)

Possible Contents: `graphics, caption?`

Description: Encloses a figure and possibly a caption, that is then placed as a (numbered) float object. If you want to refer to a graphics by an `id`, give it's `<figure>` parent element that `id`.

If you have activated two-column printing for LaTeX output, **t**book decides whether the float spans both columns or not.

Example

```
<figure id="fig:overview">
  <graphics kind="bitmap" file="overview"/>
  <caption>An overview of the whole transformation
    process.</caption>
</figure>
```

`<table>` – floating table

Attributes:

`xml:lang`	human language (inherited)
`id`	unique name (identifier)
`style/class`	style properties and class (e. g. for CSS)

Possible Contents: `tabular, caption?`

Description: Encloses a table (`<tabular>`) and possibly a caption, that is then places as a (numbered) float object. If you want to refer to a table by an `id`, give the `<table>` element that `id`.

If you have activated two-column printing for LaTeX output, **t**book decides whether the float spans both columns or not.

Example

```
<table id="tab:Matrix">
  <tabular preamble="llllll">
    <tabhead>
      <row>
<cell>Doping below/above</cell>
<cell colspan="5" align="center">Channel thickness</cell>
      </row>
      <hline from="2"/>
      <row>
<cell align="center">(<unit>10^{17} cm^{-3}</unit>)</cell>
<cell><unit>3 nm</unit></cell>
<cell><unit>2.5 nm</unit></cell>
<cell><unit>2 nm</unit></cell>
<cell><unit>1.5 nm</unit></cell>
<cell><unit>1 nm</unit></cell>
      </row>
    </tabhead>
    <tabbody>
      <srow> undoped  | T111 | T112 | T113 | T114 | T121</srow>
      <srow>  1/2     | T141 | T144 | T141 | T143 | T163</srow>
      <srow>  2/4     | T161 | T162 | T164 | T151 | T152</srow>
      <srow>  4/8     | T153 | T154 | T171 | T172 |     </srow>
      <srow>  8/16    | T192 | T191 | T193 | T194 | T214</srow>
    </tabbody>
  </tabular>
  <caption>Overview of all essential samples that have been
    prepared.</caption>
```

```
</table>
```

This yields: (In real life it will look more pleasant.)

Doping below/above	Channel thickness				
(10^{17} cm^{-3})	3 nm	2.5 nm	2 nm	1.5 nm	1 nm
undoped	T111	T112	T113	T114	T121
1/2	T141	T144	T141	T143	T163
2/4	T161	T162	T164	T151	T152
4/8	T153	T154	T171	T172	
8/16	T192	T191	T193	T194	T214

<graphics>

Attributes:

`file`	filename *without extension* (required)
`scale`	scaling factor
`kind`	type of source image, `"vector"`, `"bitmap"`, `"overlay"` or `"diagram"` (required)
`basefontsize`	assumed font size within the graphics (default: same as in document)
`xml:lang`	human language (inherited)
`id`	unique name (identifier)
`style/class`	style properties and class (e. g. for CSS)

Possible Contents: `psfrag*`

Description:

This includes a graphics that is taken from `file` (without file name extension). `kind` is interpreted as follows:

`"vector"` is an EPS file with correct bounding box.

`"bitmap"` is a JPEG bitmap with correct dpi resolution information.

`"overlay"`
is a JPEG bitmap like `"bitmap"` with an equally big EPS vector image that is printed over the bitmap as a label layer. The EPS file has the file name `file` plus an ‘`l`’.

`"diagram"`
is a LaTeX fragment read in directly. It may be e. g. Gnuplot output.

Eventually the XML processor must see how to interpret `kind`. I explain here the way the current tbook tools go.

`basefontsize` may be `"10pt"`, `"11pt"` or `"12pt"`. Sometimes one changes the global font size in a document which may make all Psfrag labels look ugly, namely too big or too small. Or one graphics migrate from one document to another with a different main font size. With `basefontsize` you can switch locally to the old font size. Of course, you can also use this attribute to change the label size for a certain graphics.

Example

```
<graphics kind="overlay" file="wafer1">
  <psfrag tag="GaAs"><ch>GaAs</ch></psfrag>
  <psfrag tag="AlAs"><ch>AlAs</ch></psfrag>
  <psfrag tag="4mu"><unit>4 &micro;m</unit></psfrag>
  <psfrag tag="top layer"/>
</graphics>
```

`<caption>` – figure/table caption

Attributes:

`xml:lang`	human language (inherited)
`id`	unique name (identifier)
`style/class`	style properties and class (e. g. for CSS)

Possible Contents: `(`*`Text`*` | `*`inline element`*`)*`

Description: The caption of a float object or the origin of an aphorism. If you use it within a float, i. e. a `<figure>` or a `<table>`, you can leave it empty; in this case the float only gets a number. It doesn't get a number just because it has an `id`!

Example

See [`<figure>`], page 61 and [`<table>`], page 62.

`<psfrag>` – graphics label replacement

Attributes:

`tag`	text tag in the eps file (required)
`number`	is it a number? (`"true"`/`"false"`)
`contrast`	`"boxed"` for white label background. `"inverse"` for white text colour.
`relsize`	`"large"`/`"small"`. default: `"normal"`.
`align`	alignment: `"left"` (default), `"right"`, `"center"`, `"ccenter"`
`interval`	automatic number generation
`xml:lang`	human language (inherited)
`id`	unique name (identifier)
`style/class`	style properties and class (e. g. for CSS)

Possible Contents: `(`*`Text`*` | `*`inline element`*`)*`

Description: The `tag` is searched in the surrounding vector graphics and the contents of the `<psfrag>` is substituted for it:

```
<psfrag tag="x-axis"><m>x</m>-axis</psfrag>
```

For this I use of course the fantastic Psfrag package. Is `<psfrag>` empty, `tag` is substituted for itself, which means that only font and size is adjusted to the main document:

```
<psfrag tag="Diagram"/>
```

If you want to erase something from an image, you have to replace it with whitespace:

```
<psfrag tag="was dull"> </psfrag>
```

`align` determines alignment relatively to the replaced `tag`:

`"left"`	left on the same baseline.
`"right"`	right on the same baseline.
`"center"`	centred on the same baseline.
`"ccenter"`	horizontally and vertically centred.

`number` is `"true"` by default, if `tag` is obviously a number, or if `interval` is given, else `"false"`.

`contrast="boxed"` sets the substitution on a white rectangle. `contrast="inverse"` shows the substitution in white colour. One of both may be necessary for too dark/chaotic backgrounds.

`relsize` should be clear.

`interval` consists, if given, of three semicolon separated numbers: start, end and step. Thus `interval="0;10;1"` yields automatically Psfrag substitutions for all numbers between 0 and 10. This is very convenient for EPS files with a labeled axis. By the way, `tag` plays in this case the role of a pattern for the numbers in the EPS file. This works somehow according to the DecimalFormat routine of Java 1.1, if anybody knows this.

Some examples:

```
<psfrag tag="#" interval="1;10;2"/>
```

replaces 1, 3, 5, 7 and 9 by itself (i. e., it changes the font only). Now for something more complicated:

```
<psfrag tag="#.0" interval="-4.5;-6;-0.5">#,0</psfrag>
```

replaces `-4.5`, `-5.0`, `-5.5` and `-6.0` by the same numbers, but with a comma instead of a point (for our non-English friends). Additionally,

```
<psfrag tag="#.0" interval="-4.5;-6;-0.5">#,#</psfrag>
```

does the same, but in the output post-comma digits (and the comma) are omitted where they are zero anyway. Last example:

```
<psfrag tag="0.0" interval="-1.5;1;0.5">#,#</psfrag>
```

does the following substitutions: `-1.5` ⇒ −1,5, `-1.0` ⇒ −1, `-0.5` ⇒ −0,5, `0.0` ⇒ 0, `0.5` ⇒ 0,5 and `1.0` ⇒ 1.

Besides, it doesn't do any harm if you declare too many replacements, so you can use the same `<psfrag>` for all your diagrams, for example.

9.14 Tables

`<tabular>` – table

Attributes:

`preamble`	LaTeX style tabular preamble
`xml:lang`	human language (inherited)
`id`	unique name (identifier)
`style/class`	style properties and class (e. g. for CSS)

Possible Contents: `tabhead?, tabbody`

Description: The `preamble` looks like a simple LaTeX tabular preamble. `"lcc"` means "three columns, one left aligned, then two centred". Except `l`, `r` and `c` nothing is allowed (yet), in particular no vertical bars, because in very most cases they are bad style.

Example

See [`<table>`], page 62.

`<tabhead>` – headline of a table

No attributes.

Possible Contents: `(hline | row | srow)*`

Description: Here every column gets a description.

Example

See [`<table>`], page 62.

`<tabbody>` – main matter of a table

No attributes.

Possible Contents: `(hline | row | srow)*`

Description: Contains the actual data rows of a table.

Example

See [`<table>`], page 62.

`<row>` – table row

Attributes:

`xml:lang`	human language (inherited)
`id`	unique name (identifier)
`style/class`	style properties and class (e. g. for CSS)

Possible Contents: `cell*`

Description: One row in a table with explicit `<cell>`s.

Example

See [`<table>`], page 62.

`<srow>` – table row with simple syntax

Attributes:

`xml:lang`	human language (inherited)
`id`	unique name (identifier)
`style/class`	style properties and class (e. g. for CSS)

Possible Contents: *`Text`*

Description: One table row. The columns are separated by '`|`' characters. This is intended for simple rows that don't need special formatting. It saves you from typing this `<cell>...</cell>` stuff.

Example

See [`<table>`], page 62.

`<hline>` – horizontal rule in a table

Attributes:

`from`	begin column (default: `"1"`)
`to`	end column (default: last)
`trim`	`"lr"`, `"l"`, `"r"` or `"no"` – trimming of rule ends

Possible Contents: None.

Description: Inserts a horizontal line in a table. `from` is the starting column, `to` the ending column. By default, a line spans the whole width. Also by default, a line ending is trimmed (shortened) if the line ends *within* the table. The `trim` can change this. `"lr"` means “trim left and right”, `"l"`, `"r"` accordingly, and `"no"` means “keep full length under all circumstances”.

The three standard rules of LaTeX’s booktabs package are always present and mustn’t be given explicitly.

Example

See [`<table>`], page 62.

`<cell>` – table entry

Attributes:

`colspan`	number of columns the entry occupies (default: `"1"`)
`align`	`"left"`, `"center"` or `"right"` – alignment (default: value from preamble)
`xml:lang`	human language (inherited)
`id`	unique name (identifier)
`style/class`	style properties and class (e. g. for CSS)

Possible Contents: `(`*`Text`* `|` *`inline element`*`)*`

Description: Encloses one rectangular row/column field in a table. `colspan` corresponds to the `\multicolumn` macro in LaTeX, and `align` is clear, I think.

Example

See [`<table>`], page 62.

9.15 Elements of the frontmatter

`<title>` – title of the document

Attributes:

`xml:lang`	human language (inherited)
`id`	unique name (identifier)
`style/class`	style properties and class (e. g. for CSS)

Possible Contents: `(`*`Text`* `|` *`inline element`* `| newline)*`

Description: Title of the document. Appears on the title page and in the title bar of the browser window.

Example

See [`<article>`], page 38 and [`<frontmatter>`], page 37.

`<author>` – full name of one author

Attributes:

`xml:lang`	human language (inherited)
`id`	unique name (identifier)
`style/class`	style properties and class (e. g. for CSS)

Possible Contents: `(`*`Text`*` | newline | footnote)*`

Description: Name of *one* author. It is automatically split up into first- and lastname, so that it can be mirrored for some cases. The point between first- and lastname is normally the *last* space, but if you give an explicit '`|`', this is used for the distinction.

`<newline>` and `<footnote>` are interpreted only for `<article>`, for `<book>` only the first text node is used.

Example

See [`<article>`], page 38 and [`<frontmatter>`], page 37.

`<subtitle>`

Attributes:

`xml:lang`	human language (inherited)
`id`	unique name (identifier)
`style/class`	style properties and class (e. g. for CSS)

Possible Contents: `(`*`Text`*` | `*`inline element`*` | newline)*`

Description: Works like the title and is used for the title page.

Example

See [`<frontmatter>`], page 37.

`<typeset>` – the artist

Attributes:

`xml:lang`	human language (inherited)
`id`	unique name (identifier)
`style/class`	style properties and class (e. g. for CSS)

Possible Contents: *`Text`*

Description: Name of the typesetter and maybe also the used font, program (TEX) etc.

Example

See [<frontmatter>], page 37.

<date> – of print

Attributes:

`xml:lang`	human language (inherited)
`id`	unique name (identifier)
`style/class`	style properties and class (e. g. for CSS)

Possible Contents: *`Text`*

Description: Date of print. The format is free, i. e. it will we printed unchanged as you've give it here.

Example

See [<article>], page 38, [<frontmatter>], page 37, and [<letter>], page 39.

<keywords>

Attributes:

`xml:lang`	human language (inherited)
`id`	unique name (identifier)
`style/class`	style properties and class (e. g. for CSS)

Possible Contents: *`Text`*

Description: Keywords describing the document. This element is exclusively used for meta information. For HTML, it creates `<meta>` elements, in PDF files it's used for the Acrobat Reader "Summary" field.

Example

See [<article>], page 38 and [<frontmatter>], page 37.

`<year>` – of copyright

Attributes:

`id`	unique name (identifier)
`style/class`	style properties and class (e. g. for CSS)

Possible Contents: *`Text`*

Description: Year(s) for the copyright. If you omit it, the **t**book stylesheets use the current year.

Example

See [`<frontmatter>`], page 37.

`<city>`

Attributes:

`xml:lang`	human language (inherited)
`id`	unique name (identifier)
`style/class`	style properties and class (e. g. for CSS)

Possible Contents: *`Text`*

Description: City of coming into existence. Also for the copyright. For a letter, this is the city printed directly before the date in the header.

Example

See [`<frontmatter>`], page 37.

`<legalnotice>`

Attributes:

`xml:lang`	human language (inherited)
`id`	unique name (identifier)
`style/class`	style properties and class (e. g. for CSS)

Possible Contents: `(`*`Text`*` | `*`inline element`*` | p)*`

Description: A disclaimer or things like that. If you give it as an empty element `<legalnotice/>`, a default legal notice is used, which you may not agree with.

Example

See [`<frontmatter>`], page 37.

`<abstract>` – summary of an article

Attributes:

`xml:lang`	human language (inherited)
`id`	unique name (identifier)
`style/class`	style properties and class (e. g. for CSS)

Possible Contents: `p+`

Example

See [`<article>`], page 38.

9.16 References

`<references>` – list of cited material

Attributes:

`bibfile`	BibTeX filename
`xml:lang`	human language (inherited)
`id`	unique name (identifier)
`style/class`	style properties and class (e. g. for CSS)

Possible Contents: *`block element*`*

Description: Inserts a references list. The contents of this element is printed as a preamble to it.

`bibfile` denotes the bibliography file. A possibly included file name extension is ignored. The default is the value of `bib-filename`, a parameter that can be given to the XSLT stylesheet when calling the XSLT processor. If even that is not given, `"biblio"` is used.

Example

See [`<article>`], page 38 and [`<backmatter>`], page 38.

9.17 Index

`<index>`

Attributes:

`xml:lang`	human language (inherited)
`id`	unique name (identifier)
`style/class`	style properties and class (e. g. for CSS)

Possible Contents: *`block element*`*

Description: Inserts an index. The contents of this element is printed as a preamble to it.

Example

See [`<backmatter>`], page 38.

`<ix>` – index entry

Attributes:

`sortkey`	sorting key
`kind`	`"emph"`, `"bold"`, `"italic"`, `"start"` or `"end"`
`xml:lang`	human language (inherited)

Possible Contents: `(`*`Text`*` | `*`inline element (except footnote, cross reference, and index entry)`*` | ix2)*`

Description: One index entry. There mustn't be more than one `<ix2>` element within `<ix>`, and that must come last.

You don't have to watch out for special characters. Everything is escaped if necessary. Most Latin letters with diacritic symbols are sorted properly.

The optional `<ix2>` contains the sub-entry, i. e. the second level. To sum it up, the old MakeIndex commad

```
\index{S"anger!Twopac@2~Pac|emph}
```

looks in **t**book like this:

```
<ix kind="emph">Sänger<ix2 sortkey="Twopac">2 Pac</ix2></ix>
```

(Sänger = singer in German.)

`<ix2>` – index sub entry

Attributes:

`sortkey`	sorting key

Possible Contents: `(`*Text* `|` *inline element (except footnote, cross reference, and index entry)*`)*`

Description: See `<ix>` above.

Example

See [`<ix>`], page 73.

`<idx>` – index entry with insertion

Attributes:

`sortkey`	sorting key
`kind`	`"emph"`, `"bold"`, `"italic"`, `"start"` or `"end"`
`xml:lang`	human language (inherited)

Possible Contents: `(`*Text* `|` *inline element (except footnote, cross reference, and index entry)*`)*`

Description: Does the same as `<ix>`, but aditionally inserts its contents at the current text position.

Example

See [`<ix>`], page 73.

`<indexsee>` – cross reference within index

No attributes.

Possible Contents: `ix, ix+`

Description: This creates cross references within the index. Apparently it has to contain at least two `<ix>` elements. The last in the row is allways the one all the others are pointing to.

9.18 letter elements

`<to>` – recipient

Attributes:

`nickname`	ID of the recipient in the address book file.
`xml:lang`	human language (inherited)
`id`	unique name (identifier)
`style/class`	style properties and class (e. g. for CSS)

Possible Contents: `(`*`Text`*` | `*`inline element`*` | newline)*`

Description: Encloses the recipient. Single lines can be separated with `<newline/>`s. If the `nickname` attribute is given *and* there is contents, the contents of this element has higher priority. So, if you want to use the address book, leave it empty:

```
<to nickname="Knuth"/>
```

Example

See [`<letter>`], page 39.

`<subject>`

Attributes:

`silent`	`"true"` or `"false"`
`xml:lang`	human language (inherited)
`id`	unique name (identifier)
`style/class`	style properties and class (e. g. for CSS)

Possible Contents: `(`*`Text`*` | `*`inline element`*`)*`

Description: The subject of a letter. This element is mandatory, but is suppressed for informal letters (attribute `formal="false"`). For formal letters, it's printed. With `silent` you may change that behaviour explicitly.

Example

See [`<letter>`], page 39.

`<opening>` – begin of letter

Attributes:

`xml:lang`	human language (inherited)
`id`	unique name (identifier)
`style/class`	style properties and class (e. g. for CSS)

Possible Contents: `(`*`Text`*` | `*`inline element`*`)*`

Description: Corresponds to `\opening{...}` in LaTeX. It begins the letter text.

Example

See [`<letter>`], page 39.

`<closing>` – end of letter

Attributes:

`kind`	`"above"`, `"below"` or `"signature"`
`xml:lang`	human language (inherited)
`id`	unique name (identifier)
`style/class`	style properties and class (e. g. for CSS)

Possible Contents: `(`*`Text`*` | `*`inline element`*`)*`

Description: It ends the letter with "Sincerely yours . . ." or something like that.

The default for `kind` is `"signature"` for formal letters and `"above"` else. `"above"` means that the contents of `<closing>` is printed above the space for the signature, `"below"` means, well, below, and `"signature"` means above, but with the name of the author below.

So, if you only want to have your name below the signature, you have to say:

```
<closing kind="signature"/>
```

Example

See [`<letter>`], page 39.

10 Known problems with tbook and associated applications

10.1 Associated applications

Unfortunately tbook has to rely on many external applications which means that any bugs in those applications effectively are bugs in tbook, too.

10.1.1 Saxon 6.5.2

Saxon is the XSLT processor used in tbook. It is called by '`tbtolatex`', '`tbtohtml`', and '`tbtodocbk`'.

1. High Unicode numbers (> 65535) cannot be processed in most places. Some people want to use e. g. Fraktur letters in their formulae. The first problem they experience is that the standard MathML2 entity names (e. g. `&Mfr;`) are not defined in tbook. I excluded them because Saxon can't process them in many cases. This is a known Saxon bug. See the Saxon bug tracker for the whole story.

10.1.2 Netscape/Mozilla

Mozilla uses a very big bug tracking system called Bugzilla. Despite it it's unfortunately very rare that non-fatal bugs are fixed. But although the following list seems to be long, the problems are small by and large.

1. Unicode spacing characters are printed as symbols. This is a known Mozilla bug but as long as Mozilla is reluctant to solve it, tbook silently substitutes ordinary spaces for those Unicode symbols. This includes `&thinsp;` (Thin Space), `&emsp;` (Em Space), `&zwnj;` (Zero Width Non-Joiner for breaking up ligatures), and `&bph;` (Break Permitted Here).
2. Stacking of equations does not work perfectly. The MathML tag `<mlabeledtr>` is necessary if you use stacked equations with numbering. Unfortunately Mozilla ignores this tag and prints such equations next to each other, so tbook uses `<mtr>` tags instead as a provisional solution. However this means that the position of the equation number is not optimal.
3. Bitmap equations may cause bad line breaks. This is a known Mozilla bug. Immediately before and after an inline bitmap Mozilla may insert a line break even if there is no space character. A typical (German) example is "x-Achse" (x axis). If the "x" becomes a small PNG bitmap, it may look in Mozilla like this:
   ```
   blah blah blah blah blah blah blah x
   -Achse blah blah blah blah blah
   ```
   which is of course unacceptable.
4. Bad line breaks with MathML equations. The MathML code in Mozilla is still under development and particularly the line breaking algorithm is all but mature. Please note that all parts embraced by `<mrow>` (or `{}` in LaTeX syntax) will never be broken, neither in HTML nor in LaTeX. So if you write `<m>{E=mc^2}</m>` this whole equation will always be on the same line.

   There are also bad line breaks *around* a MathML equation, similar to the problems with small bitmap equations above. This, too, is a known Mozilla bug.

5. Links to equations doesn't seem to work. When you click on a link to an equations, nothing happens. This is a known Mozilla bug. Mozilla doesn't know that ID attributes at MathML tags are real IDs that it can jump to. The underlying reason is that it doesn't read a proper MathML DTD. You can embed the necessary attribute declarations manually in your XHTML files like here:

   ```
   <!DOCTYPE html
     PUBLIC "-//W3C//DTD XHTML 1.1 plus MathML 2.0//EN"
            "http://www.w3.org/TR/MathML2/dtd/xhtml-math11-f.dtd"
   [
     <!ATTLIST mtd id ID #IMPLIED>
     ... the same for other MathML elements ...
   ]>
   ```

   tbook does so by itself if you call it with the undocumented XSLT parameter '`mathml-ids-in-doctype`' set to '`true`'. However, the Internet Explorer is stupid enough to display the last "`]>`". Therefore tbook uses slightly different CSS code in this case, hiding the "`]>`" by setting the text colour to white. This works at least for Mozilla and IE 6, but please test it with other browsers, too, before you use it. Feedback concerning this is welcome.[1]

10.1.3 Internet Explorer and MathPlayer

1. Support for CSS in Explorer is full of nasty bugs. In particular, it may be that you see a horizontal scrollbar because the viewframe is wider than the screen although this shouldn't be necessary. The text justification may be broken. The document title may use a too large font.
2. Some Unicode characters are not displayed. Explorer's Unicode support is worse than that of other modern browsers.
3. Practically no XML support. Although Microsoft has co-produced the XML specification, their browser is not able to show such files. Therefore all XHTML files must be served as ordinary HTML files. See [Hickson's trick], page 14, for how you can achieve this.
4. Some equations vanish. If a symbol is used that can't be found in the fonts installed on the system, MathPlayer doesn't print the whole formula at all. I found this bug e. g. with "$\hbar$" and the vector arrow. However it seems to depend on the individual system.

Please also consult the list of known Explorer issues and the list of known MathPlayer issues at Design Science (the creators of the MathML plug-in for Internet Explorer).

10.1.4 Konqueror and Safari

(The following is probably also true for Mac OSX's Safari web browser since it uses the same code.)

Konqueror has buggy CSS support. This is not very special since all browsers fail to support CSS fully. However its kind of failure makes it impossible to work with demanding CSS code optimised for Internet Explorer and Mozilla's Gecko engine.

[1] Since 2003/10/22 this bug is fixed in sources. The upcoming versions (that will contain this fix presumably) are Mozilla 1.6 and Firebird 0.7.

The exact problem of Konqueror is that floats (tables and figures) are overprinted by the text that is supposed to wrap around them.

In short, you have two possibilities: Ignore Konqueror and its users and wait for a better Konqueror version, or use the '`css-mode`' XSLT option with '`very-careful`', see Section 4.1.1 [Further XSLT parameters], page 18. For example,

```
tbtohtml mybook css-mode=very-careful
```

10.1.5 Jade

Jade is used for validating tbook documents without converting them (e. g. Emacs calls Jade to test whether the syntax of the current tbook buffer is valid). It is also used to convert DocBook documents, which means that is converts tbook to RTF for example.

1. Jade reads an invalid declaration file. All SGML implementations rely on an erroneous '`xml.decl`' file. See Section 3.6 [Jade/nsgmls issues], page 14, for how to fix this.
2. Jade complains about invalid `xmlns:...` attributes. Jade isn't aware of XML's namespace conventions. You can ignore these messages completely.
3. Jade complains about invalid MathML elements. The standard DocBook XML DTD doesn't contain MathML, so this is only natural. Since there is no DocBook tool known to me that can actually process MathML anyway, I consider this a somebody else's problem, too.

10.1.6 HTML validators

tbook's (X)HTML files will almost always be validated by web validators. However, sometimes the W3 validator claims that attributes in tbook's HTML4 output contain invalid characters. This is a limitation of XSLT 1.0 processors, and it only occurs if you use non-ASCII characters in ID and URL attributes. However, it is not possible to use XSLT 2.0 processors at the moment, for various reasons.

tbook prints a warning message in such cases. May may choose a less dangerous attribute value. On the other hand, those validation errors are absolutely nothing to worry about anyway.

10.2 List of wishes for tbook

The following things would be very nice for an upcoming tbook release. If anyone wants to help with this, please contact the tbook maintainer!

- A new program called '`tbmake`' that replaces all other programs. It would work a little bit like MikTEX's '`texify`' or a classical '`make`' implementation. It would do everything that is necessary to bring the current document up to date, including index, bibliography, and graphics in all needed formats.
- Better support for AMSTEX features, in particular its very good abilities to align stacked equations.
- Switch to the KOMA LATEX document classes. This would make configurability even better and make everything more homogeneous.
- Support ConTEXt as another output format.
- Support OpenOffice XML file format as another output format.

- Make **t**book an XSLT 2.0 application that doesn't use any Saxon specific code anymore.
- Different "skins", i. e. nice layout styles for LaTeX, and CSS for HTML.
- Better meta data in the head of an **t**book document. Author, date, keywords etc. are not enough. It may make sense, too, to include XSLT parameters in the tbook file, although the XSLT processor may ignore them.
- Allow for further image kinds, e. g. `screenshot`s. Typically screenshots will be PNGs, so they are stored in a lossless bitmap image format. SVG may be an interesting extension, too.

11 tbook source code documentation

11.1 The DTX files

A lot of tbook code is documented by DTX files. The command `make doc` transforms them into PDF files. ‘`tbookdtd.pdf`’ is the best starting point; it contains succinct documentation of the tbook DTD, although the description of elements you can also find in section Chapter 9 [Elements reference], page 35. ‘`tbookxsl.pdf`’ contains the documented source code of the XSLT stylesheets. But that file and even more the other PDFs that used to be DTX files have very poor documentation/source ratio.

(If you comment out the `\OnlyDescription` in ‘`tbookdtd.dtx`’, you get a list with most available entities and their LaTeX realisation.)

11.2 CWEB files

Two programs, ‘`tbrplent.w`’ and ‘`tbcrent.w`’, are CWEB files and thus can produce their own documentation by applying `cweave` to them. But `make doc` may have done this already.

11.3 If you can read German . . .

. . . have a look at ‘`ltxmleb.pdf`’. It contains info that may be better to digest, has some facts about the adventure of creation, and offers a good references list. It doesn’t contain documentation about tbook that’s not documented in English elsewhere in the distribution.

Index

All tbook elements are printed in `typewriter` style.

A

`abstract` 72
acronyms 45
adr-filename (XSLT parameter) 17
AMSTeX 79
anti-aliasing (XSLT parameter) 17
`aphorism` 58
`appendix` 43
array (LaTeX Package) 3
`article` 39
assume-os (XSLT parameter) 17
`author` 69
availability 2

B

babel 3
`backmatter` 38
bib-filename (XSLT parameter) 17
bibfix 33
bibliography 10, 11, 33, 47, 72
BibTeX 1, 33
BibTeX, styles 4
BINDIR 7
bitmaps 63
block elements 35
`blockquote` 56
bold 45
`book` 36
booktabs 3, 68
braces, horizontal 28
browser 13
browsers, older 14
bug reports 2
bugs, known 77

C

`caption` 64
caption2 3
cat (XSLT parameter) 18
`cell` 68
cfg file 21
`ch` 25, 50
`chapter` 41
chemical formulae 50
citations 47
`cite` 47
`city` 71
class attribute 31
`closing` 76
colour 31
columns 18
concept, general 1, 9
configurability 17
content-type 14
ConTeXt 79
conversions 9
CookTop 7
cp (XSLT parameter) 18
create-image-comments (XSLT parameter) 17
cross references 46, 47, 49
CSS 1, 20, 31
css-file (XSLT parameter) 17
css-mode (XSLT parameter) 19
CTAN 3
customisation layers 22
CVS tree 2
CWEB 4, 81

D

`date` 70
debug (XSLT parameter) 19
declaration file 14
`description` 60
desperate-measures (XSLT parameter) 17
diagram 13
`dm` 25, 50
DocBook 1, 9
docbook-equations (XSLT parameter) 19
document-fontsize (XSLT parameter) 17
downloads 2
driver files 22
DTD 1, 9
DTX files 81
dvips 10
dvips-offset (XSLT parameter) 19

E

edition (field of BibTeX) 33
editors 1, 7
elements reference 35
`em` 45
Emacs 9
`enumerate` 59
epigraph 58
EPS graphics 10
eqn-gallery-filename (XSLT parameter) 19
equation arrays 26
equation-resolution (XSLT parameter) 19
equations 25, 49
equations, bitmaps 10
equations, numbering 27

example 11
example, minimal 35
experience report 81
Explorer 78

F

fancyhdr 3, 22
FAQ 2
features 1
features, upcoming 79
`figure` 61
figures 63
file name database 4
file-extension (XSLT parameter) 19
Flex 4
floats 35, 61
font change 45
fontenc 3
fontsize 17
`footnote` 45
formulae 17, 25, 49
formulae, numbering 27
`frontmatter` 37

G

gallery-filename (XSLT parameter) 19
general concept 1, 9
geometry (LaTeX package) 3, 22
German documentation 81
Ghostscript 4, 19
Ghostscript, download 6
given-document-name (XSLT parameter) 19
glossary 60
Gnuplot 13
graphics 61
`graphics` 63
graphics, files 13
graphics-fontsize (XSLT parameter) 19
gs (XSLT parameter) 19

H

`heading` 40
help, general 3
helvet 3
`hline` 67
homepage 2
horizontal rule 67
horizontal skip 53
`hspace` 53
HTML 1, 9, 13
html-equations (XSLT parameter) 17
html-extension (XSLT parameter) 19
html-resolution (XSLT parameter) 18
human languages 1, 3, 29
hypcap 3
hyperref 3

I

IANA 30
`idx` 74
IE 78
include-image-dimensions (XSLT parameter) 18
include-originals (XSLT parameter) 19
index 10, 11
`index` 73
index-at (XSLT parameter) 19
index-input-filename (XSLT parameter) 19
index-markup-separator (XSLT parameter) 19
index-output-filename (XSLT parameter) 19
index-separator (XSLT parameter) 19
`indexsee` 74
inline elements 35
install.bat 7
installation 3
installation, Linux 3
installation, Windows 6
Internet addresses 2
Internet Explorer 14, 78
invocation 9
invoking 9
italic 45
`item` 61
`itemize` 59
`ix` 73
`ix2` 74

J

Jade 9, 14, 79
Java 3
JDK 3
JPEG graphics 10
jpeg-quality (XSLT parameter) 18
jpeg2ps 4

K

`keywords` 70
KOMA 79
Konqueror 78

L

languages, human . . . 1, 3, 29
latex . . . 10
latex . . . 54
LaTeX . . . 1, 9
LaTeX math syntax . . . 49
LaTeX packages . . . 3
LaTeX style file . . . 22
latex-quiet-option (XSLT parameter) . . . 20
lb (XSLT parameter) . . . 20
legalnotice . . . 71
letter . . . 39
Linux installation from sources . . . 3
Linux, installation . . . 3
lists . . . 59
ltxmleb.pdf . . . 81

M

m . . . 25, 26, 49
mailing list . . . 2
mainmatter . . . 37
make scripts . . . 9
make scripts, dependencies . . . 11
makebib . . . 10
makeclean . . . 10
makeepss . . . 10
makeeqns . . . 10
Makefile . . . 3, 4
makeidx . . . 10
makepage . . . 10
makepdfs . . . 10
MakeTitlePage (LaTeX command) . . . 22
makewebs . . . 10
manpages . . . 9
markup, visual . . . 45, 53
marvosym . . . 3
math . . . 25, 26
MathML . . . 1, 13, 14, 25, 26
MathPlayer . . . 78
mathplayer-pi-location (XSLT parameter) . . . 20
mathptmx . . . 3
mathref . . . 49
maths . . . 25, 49
maths, numbering . . . 27
maths, over- und underbraces . . . 28
maximal-alt-length (XSLT parameter) . . . 18
MikTeX . . . 6, 7
MIME type . . . 14
minimal example . . . 35
minitoc . . . 3
mkdir (XSLT parameter) . . . 20
mktexlsr . . . 4
mlabeledtr . . . 27
Mozilla . . . 14, 77
mrow . . . 27
mtable . . . 26
mtd . . . 26
mtr . . . 26
multipar . . . 44
mv (XSLT parameter) . . . 20

N

natbib . . . 3
Netscape . . . 14
Netscape 6/7 . . . 77
newline . . . 44
nicefrac . . . 3
nsgmls . . . 14
NSIS . . . 6
numbering of equations . . . 27

O

OnlyDescription (LaTeX command) . . . 81
opening . . . 76
Openjade . . . 9, 14
OpenOffice . . . 79
output formats . . . 1, 9
overbrace . . . 28
overlay . . . 13

P

p . . . 43
packages, LaTeX . . . 3
pageref . . . 47
paragraph . . . 42
parameters, XSLT . . . 17
part . . . 41
patch . . . 4
PATH . . . 7
PDF . . . 1, 9
PDF graphics . . . 10
pdfcolmk . . . 3
pdflatex . . . 10
physical quantity . . . 54
pictures . . . 63
pifont . . . 3
PNG . . . 80
PNG graphics . . . 10
pnmfile . . . 4
pnmtopng . . . 4
Postscript . . . 1, 9
ppmtojpeg . . . 4
preview-latex (XSLT parameter) . . . 18
problems, known . . . 77
proof . . . 52
psfrag . . . 1, 3, 10
psfrag . . . 64

Q

quantity, physical 54
`quote` 55

R

ragged2e 3
`ref` 46
reference of all elements 35
references 10, 11, 33, 47
`references` 72
`relax` 53
relsize (LaTeX package) 3
rem (XSLT parameter) 20
rm (XSLT parameter) 20
root of document 36, 39
`row` 67
RPMs 3
RTF 1, 9
rule, horizontal 67

S

Safari 78
sans serif 45
Saxon 3, 4, 77
Saxon 7 15
Saxon, download 6
Saxon, manual setup 4
scripts 9
scripts, dependencies 11
secnumdepth (XSLT parameter) 18
`section` 42
sectsty 3
sed 4
server, web 14
shell scripts 9
shell scripts, dependencies 11
shift-equations (XSLT parameter) 20
skip, horizonal 53
slanted 45
small caps 45
soul (LaTeX package) 3
source code 81
span 55
split-level (XSLT parameter) 18
`srow` 67
sty-file (XSLT parameter) 18
style attribute 31
style file for LaTeX 22
`subject` 75
`subparagraph` 43
`subsection` 42
`subsubsection` 42
`subtitle` 69
SVG 80

T

`tabbody` 66
`tabhead` 66
`table` 62
tables 66
tables, rules 67
`tabular` 66
tbcrent 81
TBLIBDIR 7
tbmake 79
tbook-pl.sty 22
tbookdtd.dtx 81
tbprepare 9
tbrplent 81
tbtodocbk 10, 17
tbtohtml 9, 13, 17
tbtolatex 9, 17
`term` 61
texhash 4
texify 79
TeXLive 6
`theorem` 51
`title` 68
`to` 75
tocdepth (XSLT parameter) 18
transformations 9
transparent-pngs (XSLT parameter) 18
trouble 3
two-columns (XSLT parameter) 18
two-side (XSLT parameter) 20
`typeset` 69
typewriter 45

U

UMSS 14
underbrace 28
`unit` 25, 54
`url` 52
url (LaTeX package) 3
usage 9

V

validation 14, 79
validators (Web) 79
`verb` 56
`verbatim` 57
versaliae 45
`verse` 57
`visual` 45
visual markup 53
`vref` 47

W

wasysym . . . 3
web output . . . 13
web server . . . 14
Windows installation . . . 6
wishlist . . . 79
wordprocessors . . . 1
`wrap` . . . 55

X

Xalan . . . 15
XHTML . . . 1, 13
xindy . . . 1, 4
xindy, download . . . 7
xindy, installation . . . 3
xml.decl . . . 14
xml:lang . . . 29
XMLCookTop . . . 7
XSLT 2.0 . . . 15
XSLT parameters . . . 17
XSLT processors . . . 15

Y

`year` . . . 71