

# The **tbook** System for XML Authoring

## Version 1.4

Torsten Bronger\*

October 10, 2002

### Contents

<b>1</b>	<b>Features</b>	<b>2</b>
<b>2</b>	<b>Installation</b>	<b>3</b>
2.1	Linux RPM installation . . . . .	3
2.2	Linux installation from the sources . . . . .	4
2.3	Windows installation with the installer . . . . .	7
2.4	Windows installation from the sources . . . . .	7
<b>3</b>	<b>Usage</b>	<b>8</b>
3.1	Where to look something up . . . . .	9
3.2	Your original graphics files . . . . .	9
3.3	The shell scripts . . . . .	9
3.4	Batch file dependencies . . . . .	10
3.5	Viewing your document in a browser . . . . .	10
3.6	Jade/nsgmls issues . . . . .	11
3.7	Other XSLT processors: Saxon 7, Xalan, . . . . .	14
<b>4</b>	<b>Configurability</b>	<b>14</b>
4.1	XSLT parameters . . . . .	14
4.2	CSS style sheets . . . . .	14
4.3	L <sup>A</sup> T <sub>E</sub> X <code>cfg</code> file . . . . .	15
4.4	L <sup>A</sup> T <sub>E</sub> X <code>sty</code> file . . . . .	15
4.5	Driver files . . . . .	15
<b>5</b>	<b>tbook's “almost L<sup>A</sup>T<sub>E</sub>X” formula syntax</b>	<b>16</b>
5.1	MathML . . . . .	17
5.1.1	Equation arrays . . . . .	17
5.1.2	Equation numbers . . . . .	17

---

\*bronger@physik.rwth-aachen.de

<b>6</b>	<b>Human language support</b>	<b>18</b>
<b>7</b>	<b>Support for text colour</b>	<b>18</b>
<b>8</b>	<b>Bibliography</b>	<b>18</b>
<b>9</b>	<b>tbook source code documentation</b>	<b>19</b>
9.1	The DTX files . . . . .	19
9.2	CWEB files . . . . .	19
9.3	If you can read German . . . . .	19
<b>10</b>	<b>Elements reference</b>	<b>20</b>
10.1	Top level elements . . . . .	21
10.2	Parts, chapters and sections . . . . .	23
10.3	Paragraphs & friends . . . . .	24
10.4	Font change . . . . .	25
10.5	Cross references . . . . .	26
10.6	L <sup>A</sup> T <sub>E</sub> X-like mathematics . . . . .	28
10.7	Theorems and proofs . . . . .	29
10.8	Miscellaneous . . . . .	31
10.9	Quoted and verbatim material . . . . .	33
10.10	Lists . . . . .	34
10.11	“Floats” and their contents . . . . .	35
10.12	Tables . . . . .	38
10.13	Elements of the frontmatter . . . . .	40
10.14	Bibliography . . . . .	43
10.15	Index . . . . .	43
10.16	letter elements . . . . .	45

# 1 Features

The main features are:

- 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 L<sup>A</sup>T<sub>E</sub>X. Supported L<sup>A</sup>T<sub>E</sub>X classes: book, article and letter.
- It produces HTML, XHTML+MathML, Postscript, PDF and DocBook output.
- It works with formulas, graphics, tables, (all three with numbering) bibliography, and index. Tools that have proven their efficiency with L<sup>A</sup>T<sub>E</sub>X (BIB<sub>T</sub>E<sub>X</sub>, xindy etc.) are also usable with tbook.
- The print output caters high typographic demands, using high level L<sup>A</sup>T<sub>E</sub>X typesetting mechanisms. This includes high quality graphics and their labelling with Psfrag. Additionally, it produces small PDFs.

- It is easily configurable. For  $\text{\LaTeX}$  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 allow for direct unicode input. Supported languages so far: English, German, French, Italian, Spanish, and Catalan.
- It is possible to input XML code directly without being killed by tons of characters. (That counts twice for formulas.)
- It's a real life tool, no academic project. If it doesn't work properly for you, complain!

The project home page is at <http://tbookdtd.sourceforge.net>. There you also can find an example document, which is not included in this distribution (for saving space).

## 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  $\text{\LaTeX}$ , 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.

**tbook** needs many  $\text{\LaTeX}$  packages. Those that are part of the  $\text{\LaTeX}$  core (babel, fontenc, array, etc) are nothing to worry about, every non-ancient  $\text{\LaTeX}$  has them. The other packages are: booktabs, caption2, fancyhdr, geometry, helvet, hypcap, 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  $\text{\LaTeX}$  packages. Be aware that helvet, mathptmx, and hyperref must be as new as possible. If your versions are too old,  $\text{\LaTeX}$  will gently complain.

Every modern  $\text{\LaTeX}$  distribution contains most of them, maybe all; else  $\text{\LaTeX}$  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/supported/>.

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

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

### 2.1 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-1.4-1tb.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**.

## 2.2 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.

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

- L<sup>A</sup>T<sub>E</sub>X, pdfL<sup>A</sup>T<sub>E</sub>X, (both new and well equipped), dvips,<sup>1</sup>
- Ghostscript<sup>2</sup>, ps2pdf (part of Ghostscript),
- pnmtopng, pnmtojpeg, pnmfile,<sup>3</sup> (not *really* vital),
- **xindy**<sup>4</sup> if you need indexing,
- jpeg2ps<sup>5</sup> and
- Saxon 6.5.x<sup>6</sup> (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.

---

<sup>1</sup>all of them at <http://www.tex.ac.uk/>

<sup>2</sup><http://www.cs.wisc.edu/~ghost/doc/AFPL/get704.htm>; *don't* use GNU Ghostscript

<sup>3</sup>all three at <http://sourceforge.net/netpbm>

<sup>4</sup><http://sourceforge.net/projects/xindy>, or as an RPM from **tbook**'s project page at <http://sourceforge.net/projects/tbookdtd>

<sup>5</sup>On CTAN at [nonfree/support/jpeg2ps/](http://nonfree/support/jpeg2ps/)

<sup>6</sup><http://sourceforge.net/projects/saxon>, or as an RPM from **tbook**'s project page at <http://sourceforge.net/projects/tbookdtd>

In order to install **tbook** with some convenience, you also need CWEB<sup>7</sup>, sed, patch and Flex<sup>8</sup>. However, the Linux/Intel binaries are included in the distribution, as well as all B<sub>I</sub>T<sub>E</sub>X-Styles, if you have trouble to build them.

If these conditions 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.

This should do the following: (Unless, of course, you've changed some paths, which you probably will have to.)

1. Unpacks all files contained in the DTX files via `docstrip`.
2. Creates the B<sub>I</sub>T<sub>E</sub>X style files using the `custom-bib` package<sup>9</sup> and copies them to `/usr/local/texmf/share/texmf/bibtex/bst/tbook`.
  - The files `tb??l.bst` (with `??` being the language) are for L<sup>A</sup>T<sub>E</sub>X 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 L<sup>A</sup>T<sub>E</sub>X 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 B<sub>I</sub>T<sub>E</sub>X's XML output into (hopefully) real XML.
4. Copies these files into `/usr/local/bin`.
5. Copies the four shell scripts `tbtolatex`, `tbthtml`, `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 L<sup>A</sup>T<sub>E</sub>X input file.
  - `tbthtml` 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.

---

<sup>7</sup><http://www-cs-faculty.stanford.edu/~knuth/cweb.html>

<sup>8</sup>sed, patch and Flex are part of the GNU project at <http://www.gnu.org>

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

6. Creates the directories `/usr/local/texlive/share/texmf/tex/latex/tbook` and `/usr/local/lib/tbook`.
7. Copies the  $\text{\LaTeX}$  input files `tlpcrv.fd`, `tbook.sty` and `tbook-pl.sty` into `.../texmf/tex/latex/tbook`.
  - `tlpcrv.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  $\text{\LaTeX}$  files.
  - `tbook-pl.sty` ("**tbook** plain") is an example style package for generated  $\text{\LaTeX}$  files.
8. Copies the **tbook** XML files `tbook.dtd`, `tblatex.xsl`, `tbhtml.xsl`, `tbdocbk.xsl`, `tbtohtml.xsl`, `tbtohtml4.xsl`, `tbtdb.xsl`, `tbcommon.xsl`, `hmm12dst.dtd`, `tbents.txt` and `tbtplte.xml` to `/usr/local/lib/tbook`.
  - `tbook.dtd` is the **tbook** XML DTD.
  - `tblatex.xsl` is the XSLT for **tbook**  $\rightarrow$   $\text{\LaTeX}$ .
  - `tbhtml.xsl` is the XSLT for **tbook**  $\rightarrow$  XHTML.
  - `tbdocbk.xsl` is the XSLT for **tbook**  $\rightarrow$  XML DocBook.
  - `tbtohtml.xsl`, `tbtohtml4.xsl`, and `tbtdb.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`.
  - `hmm12dst.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.
  - `tbents.txt` is created by `tbcrent` and needed by `tbrplent`. It contains the Unicode  $\rightarrow$   $\text{\LaTeX}$  mappings.
  - `tbtplte.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  $\text{\LaTeX}$  index.
  - `tbhtml.xdy` is the **xindy** style file for an HTML index.
  - `tbook.xdy` contains shared commands for both index outputs.
10. Copies the man pages to `/usr/local/share/man/`.
11. Copies the DTD in browsable HTML format to `/usr/local/share/doc/tbook`.

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

## 2.3 Windows installation with the installer

There are also two methods to install **tbook** on the Windows operating system. The first one that should be preferred is to use the NSIS based installer/uninstaller program. Basically it's just an .EXE file called `tbook-1.4.exe` that you have to start. But before you do that, please install:

1.  $\text{\TeX}$ . Get the latest MikTeX distribution at <http://www.miktex.org> or the  $\text{\TeX}$ Live CD at <http://www.tug.org/texlive/>. The installation shouldn't be too hard.
2. Ghostscript. Get it at <http://www.cs.wisc.edu/~ghost/doc/AFPL/get704.htm> and install also Ghostview from <http://www.cs.wisc.edu/~ghost/gsview/get43.htm>, although only Ghostscript is needed by **tbook**.

The **tbook** 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 **tbook**, delete the entries in `autoexec.bat` and in the registry. To uninstall **tbook**, you may also use the standard way via system control→software.

## 2.4 Windows installation from the sources

This is an emergency solution in case that you can't (or don't want to) install **tbook** 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. *Don't install Saxon 7, but Saxon 6.5.x!*
2.  $\text{\TeX}$ . Get the latest MikTeX distribution at <http://www.miktex.org> or the  $\text{\TeX}$ Live CD at <http://www.tug.org/texlive/>. The installation shouldn't be too hard.
3. Ghostscript. Get it at <http://www.cs.wisc.edu/~ghost/doc/AFPL/get704.htm> and install also Ghostview from <http://www.cs.wisc.edu/~ghost/gsview/get43.htm>, 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  $\text{\TeX}$ , although it should. You find it at <http://sourceforge.net/projects/xindy>. 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**!

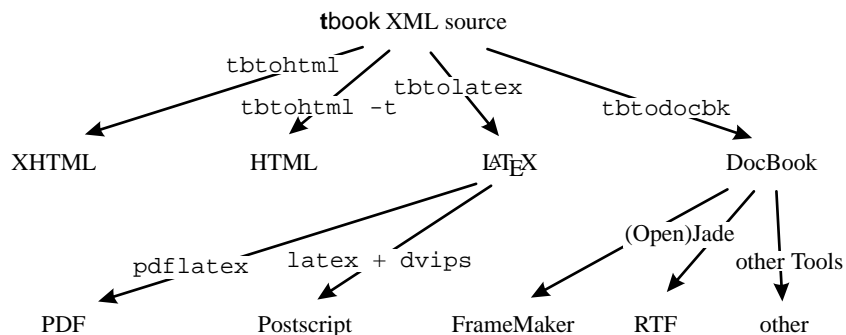


Figure 1: Possible transformations with **tbook** files.

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  $\text{T}_{\text{E}}\text{X}$ 's file database. How this is done depends on your  $\text{T}_{\text{E}}\text{X}$  implementation. With Mik $\text{T}_{\text{E}}\text{X}$  e. g., it's an item in the Mik $\text{T}_{\text{E}}\text{X}$  program group of the start bar.

That's it. Hopefully.

If you're looking for a good XML editor, have a look at <http://www.xmlcooktop.com>.

### 3 Usage

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.



### 3.1 Where to look something up

Besides this text you are reading, there are man pages available for `tbtolatex`, `tbtohtml`, `tbtdocbk`, `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/>.

### 3.2 Your original graphics files

Next to your `tbook` XML document, this directory must contain all graphics that you want to include. `tbook` 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 '1' appended ("labels").

**diagram** is a  $\text{\LaTeX}$  fragment (e.g.  $\text{\LaTeX}$  picture output by Gnuplot) with the file extension `.pic`.

### 3.3 The shell scripts

Now you have the following commands available:

- `tbtolatex <dn> [<parameters>]` creates a  $\text{\LaTeX}$  file `<dn>.tex` ready to be processed by  $\text{\LaTeX}$  or `pdf $\text{\LaTeX}$` .
- `tbtohtml <dn> [<parameters>]` creates a file `<dn>.xhtml` with an XHTML version of your document.
- `tbtdocbk <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.

Additionally, the following make files are constantly kept up-to-date with the document and create needed files:

- `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 EPSes. 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 page 12.
- `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 wild-cards, for safety.

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

`makeepss`, `makepdfs` and `makewebs` accept an optional argument which is the name of a graphics. If it's given, they process only *this* graphics. Similarly you can give `eqn-number` to `makeeqns`.

Besides that, you will have to call `LATEX`, `pdfLATEX` and `dvips` as necessary.

### 3.4 Batch file dependencies

The dependencies of the `make??` batch files and the `tbto??` shell scripts 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.

### 3.5 Viewing your document in a browser

`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 buch
```

3. Quirky HTML. This is activated with the `-t` option *and* the XSLT parameter `css-mode` set to `very-careful`:

```
tbtohtml -t buch 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 NS4 can be made happy.

Without the `-t` option, **tbook** 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<sup>10</sup> 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 `application/xhtml+xml`.

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<sup>11</sup> plugin. For this trick, have a look at <http://software.hixie.ch/utilities/cgi/xhtml-for-ie/>.

### 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 which is wrong. Try to find this file (on my system it’s called `xml.dcl`, but `xml.dcl` is also possible) and change the line

```
128      32  UNUSED
```

to

```
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 complains about namespace attributes, but this too is false alert.

---

<sup>10</sup>For free download at <http://www.mozilla.org>

<sup>11</sup><http://www.dessci.com/webmath/mathplayer>

Parameter Name	Default	Meaning
adr-filename	'adrbook.xml'	filename of the address book file
bib-filename	'biblio.xml'	filename of the $\text{BIB}\text{T}_{\text{E}}\text{X}$ file
css-file	' '	filename of a CSS file (XML format)
sty-file	'tbook-pl'	filename of a $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ sty file, '(none)' for don't use any.
document-fontsize	'10pt'	global font size. 10pt, 11pt or 12pt.
two-columns	'false'	If 'true', the document is set with two columns.
assume-os	'linux'	assumed operating system. Possible values are linux, unix, windows and os2. <sup>12</sup>
anti-aliasing	'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.
include-image-dimensions	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.
html-resolution	100	dpi resolution for HTML graphics. You can also use it for global HTML image scaling.
transparent-pngs	'false'	Should all white in PNGs become transparent? (Also applies to equation bitmaps.)
html-equations	'mathml'	With 'bitmaps' all formulae become bitmaps for HTML.
preview-latex	'false'	'true' makes the equation bitmaps generating process <i>much</i> faster. <sup>13</sup>
split-level	'none'	If 'chapter', the HTML file is divided into file chunks for every chapter.
jpeg-quality	75	JPEG quality of the HTML output in the usual JPEG unit between 0 and 100.
create-image-comments	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.
rm	rm -f	default is del for Windows & OS/2. File deletion command. <sup>14</sup>
maximal-alt-length	50	maximal number of characters in the alt attribute popup rectangle in HTML, if you are over the image with the mouse.
desperate-measures	false	if true, "desperate measures" are applied, see tbookdtd.dtx.

<sup>12</sup>Only linux and windows are tested, so the rest will surely fail. :- ) Well, it affects the look of the batch files being generated.

<sup>13</sup>Needs 'preview' package (<http://preview-latex.sf.net>) and more disk space.

<sup>14</sup>If you don't want that at all, set it e. g. to #.

Table 1: Possible XSLT parameters. Only the most important ones are listed.

Parameter Name	Default	Meaning
given-document-name	'tbook-temp-file'	assumed filename of the document, if the correct one can't be determined (e.g. not Saxon used).
gallery-filename	'<dn>-gallery'	file name of the image gallery $\LaTeX$ file
eqn-gallery-filename	'<dn>-eqn-gallery'	file name of the formulae gallery $\LaTeX$ file
equation-resolution	value of html-resolution	Dpi resolution of the bitmap equations.
dvips-offset	0pt,0pt	Dvips offset for the gallery processes.
include-originals	'true'	If 'true', you can click on bitmaps in HTML and get a view of the original JPEG.
graphics-fontsize	'default'	font size assumed for all graphics. Default means "same of document". 10pt, 11pt or 12pt. See "basefontsize" in tbookdtd.dtx.
file-extension	' '	file extension of generated batch files, including a possible leading dot.
rem	'# '	Command that introduces comments in batch files.
rm	'rm -f'	Shell command for file deletion.
mv	'mv -f'	Shell command for file renaming.
cp	'cp -f'	Shell command for file copying.
mkdir	'mkdir -p'	Shell command for creating a sub directory.
cat	'cat'	Shell command for sending a file to standard output.
lb	' \&#10; '	Line break within a batch file.
latex-quiet-option	'-interaction=batchmode'	How to keep $\TeX$ calm.
gs	'gs'	How to invoke Ghostscript.
index-markup-separator	' '	Symbol that introduces markup info in xindy output
index-separator	'!'	Symbol to separate xindy entry from subentry
index-at	'@'	Symbol before the sorting key. Also xindy.
index-output-filename	'<dn>-idx.xml	file name for xindy input (which is XSLT <i>output</i> .)
index-input-filename	'<dn>-ind.xml	file name for xindy output (which is XSLT <i>input</i> .)
css-mode	'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.
debug	'false'	If 'true', you see all $\LaTeX$ messages.
html-extension	'.xhtml'	Assumed file extension for the HTML file.
shift-equations	'true'	Should bitmap equations have a corrected baseline? ('false' necessary for Netscape 4.77).
docbook-equations	'text'	If 'mathml', DocBook XML+MathML is created.

Table 2: Less important XSLT parameters. The default values should be okay for most cases. Notice that many parameters have OS dependent defaults. These here are the Unix values. The full rules can be found in `tbookxsl.dtx`, chapter "Common Code", section 4.3.

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

At the moment, **tbook** 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 batch files, 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 parallelly with Saxon 7.

As soon as XSLT 2.0 is a finished specification, **tbook** will support it. Then you can use **tbook** 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

`tbtolatemx`, `tbtothtml` and `tbtotdocbk` can pass parameters to the XSLT processor Saxon in the form `<parametername>=<value>`, so e. g.

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

For a list of the available parameters, have a look at tables 1 and 2.

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

```
tbtolatemx 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!

### 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 commands. They are loaded *last* and thus have priority. You can change the final layout of your HTML document in many ways.

### 4.3 L<sup>A</sup>T<sub>E</sub>X cfg file

Is in the L<sup>A</sup>T<sub>E</sub>X 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.

### 4.4 L<sup>A</sup>T<sub>E</sub>X sty file

With the XSLT parameter `sty-file` you can include an arbitrary L<sup>A</sup>T<sub>E</sub>X 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
```

- 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 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 L<sup>A</sup>T<sub>E</sub>X 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 L<sup>A</sup>T<sub>E</sub>X 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.xml`, `tbthtml.xml`, `tbthtml4.xml`, and `tbtodb.xml`. These files are almost empty, actually they only import the really big ones `tblatex.xml`, `tbhtml.xml` and `tbdocbk.xml`, which you shouldn't modify. Such modifications 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 L<sup>A</sup>T<sub>E</sub>X" formula syntax

You can use directly MathML's presentation markup in **tbook** files. For this, insert `<math>` elements without worrying about namespaces. But except for not too complicated equations this is quite longish.

**tbook**'s formula elements `<m>` and friends use simplified L<sup>A</sup>T<sub>E</sub>X 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 L<sup>A</sup>T<sub>E</sub>X. Human text is included via `\text` that is known from AMS<sup>T</sup><sub>E</sub>X. Standard functions are typed without a `\`. Stretchable braces are all braces immediately within a `{...}` grouping. Sub- and superscripts as in L<sup>A</sup>T<sub>E</sub>X, 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>&Hat;{1-x_{\text{eff}}} &ne; {( \int_0^{\infty} sin(\tilde{x})
      \frac{\sqrt[3]{1/e}}{\beta}dx )}
      &ne; \lim_{x \rightarrow \infty} \frac{1}{x}
```

becomes

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

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. If you use these elements within a MathML equation *array*, you can generate alignment markers (in L<sup>A</sup>T<sub>E</sub>X known as '&' signs) with '#' signs, because this is shorter than `&amsp; i`. Put them where they would be in L<sup>A</sup>T<sub>E</sub>X. Although MathML requires such a marker at the very beginning of an equation row, this is not true for L<sup>A</sup>T<sub>E</sub>X, and not true for **tbook**.



## 5.1 MathML

In this context some words about MathML. You may use it if you want, for more complicated formulas you must use it, unfortunately. It's always enclosed by `<math>...</math>`, but only presentation markup can be transformed to  $\text{\LaTeX}$  yet.

### 5.1.1 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 `<mtabledtr>` rows. If you set `groupalign="right center left"`, this leads to an `eqnarray` in  $\text{\LaTeX}$ , and where `'&'` are in  $\text{\LaTeX}$ , 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:

```
<math>
  <mtable groupalign="right center left">
    <mtr>
      <mtid id="test"> <m> 1+1 \equiv 2 </m> </mtid>
      <mtid> <m> 4 \equiv 2 \cdot 2 \cdot 2 </m> </mtid>
    </mtr>
  </mtable>
</math>
```

which is the same as  $\text{\LaTeX}$ 's

```
\begin{eqnarray}
  1+1 \equiv 2 \label{Test} \\
  4 \equiv 2 \cdot 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 `<mtid>`, `<m>` is implicitly surrounded by an `<mrow>` which is necessary in this context.

### 5.1.2 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. Plain and simple.
2. An `<mtid>` element with an `id` within an equation array (see above).
3. An `<mtabledtr>` element with an `id` within an equation array, and the contents of the first `<mtid>` element of such an `<mtabledtr>` row.

You may use the `<ref>` element to refer to such equations, but for the contents of the `<mtid>` element in the third case, you have to use `<mathref>`. I would recommend you to use only 1. and 2. See the MathML specs at the W3C for more information.

## 6 Human language support

**tbook** supports the following XML language codes:

"en"	English
"de"	German
"fr"	French
"it"	Italian
"es"	Spanish
"ca"	Catalan
"en-US"	American English
"en-GB"	British English
"de-DE"	FR German
"de-AT"	Austrian
"de-1901"	German, traditional orthogr.
"de-1996"	German, new orthogr.
"de-DE-1901"	FR German, traditional orthogr.
"de-AT-1901"	Austrian, traditional orthogr.
"de-DE-1996"	FR German, new orthogr.
"de-AT-1996"	Austrian, new orthogr.

The “special” German orthography tags have been registered with IANA.

If you need another language, contact the **tbook** maintainer.

## 7 Support for text colour

There is no element that adds colour to a **tbook** 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  $\text{\LaTeX}$  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  $\text{\BIBTeX}$  file, although minor adjustments could be necessary. It is preferred, although not necessary, that if your  $\text{\BIBTeX}$  files contain accented characters or something like this, they should be 8-bit files.

**tbook's** BIB<sub>T</sub>E<sub>X</sub> styles (by the way, created with `custom-bib`) will create decent L<sup>A</sup>T<sub>E</sub>X output, but for HTML and DocBook, a filter called `bibfix` tries to make BIB<sub>T</sub>E<sub>X</sub>'s almost-XML output real XML. If you've used strange L<sup>A</sup>T<sub>E</sub>X constructs in your BIB<sub>T</sub>E<sub>X</sub> file, you either have to fix the BIB<sub>T</sub>E<sub>X</sub> file, or you have to extend `bibfix.l`, a simple flex scanner, to convert them to XML.

Another possible source of trouble in old BIB<sub>T</sub>E<sub>X</sub> files is the `edition` field. Best is to use an English number in letters like `"third"`.

## 9 tbook source code documentation

### 9.1 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 10. `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 docu/source ratio.

(If you comment out the `\OnlyDescription` in `tbookdtd.dtx`, you get a list with most available entities and their L<sup>A</sup>T<sub>E</sub>X realisation.)

### 9.2 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.

### 9.3 If you can read German ...

... have a look at `ltxmleeb.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.

## 10 Elements reference

A minimal **tbook** file looks like this:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE book PUBLIC "-//Torsten Bronger//DTD tbook 1.3//EN"
    "http://tbookdtd.sourceforge.net/tbook13.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>
```

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 refernces: `<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>`, `<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 online version of this element reference is available at <http://tbookdtd.sourceforge.net/dtd/>.

## 10.1 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  $\text{\LaTeX}$  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>`.

---

### <frontmatter> – general info about a book

**Attributes:** None.

**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.)

---

### <mainmatter> – the text contents

**Attributes:** None.

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

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

---

**<backmatter> – references and index**

**Attributes:** None.

**Possible contents:** `references?, index?`

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

---

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

**Attributes:**

<code>xml:lang</code>	human language (default: "en")
<code>id</code>	unique name (identifier)
<code>style/class</code>	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.

---

**<letter> – a letter (root element)**

**Attributes:**

<code>from</code>	unique signature of the author
<code>formal</code>	"true" for "formal letter" or "false"
<code>xml:lang</code>	human language (default: "en")
<code>id</code>	unique name (identifier)
<code>style/class</code>	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).<sup>15</sup> Moreover the `owner` attribute in the address book must match it.

---

<sup>15</sup>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.

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.

## 10.2 Parts, chapters and sections

---

### **<heading> – of a chapter etc.**

#### **Attributes:**

<code>xml:lang</code>	human language (inherited)
<code>id</code>	unique name (identifier)
<code>style/class</code>	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.

---

### **<part> – of a book**

#### **Attributes:**

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

**Possible contents:** *heading, chapter\**

**Description:** Embraces a part of a book.

---

### **<chapter> – of a book**

#### **Attributes:**

<code>kind</code>	"preface", "introduction", "acknowledgements" or "colophon" – marks chapters with special meaning
<code>xml:lang</code>	human language (inherited)
<code>id</code>	unique name (identifier)
<code>style/class</code>	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 starred 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.

---

**<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.

**<subsection>, <subsubsection>, <paragraph> and <subparagraph>**

These elements are totally analogous to `<section>`, they can contain the next lower level of structuring, respectively. `<subsection>` wanders in any case in the TOC.

---

**<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 L<sup>A</sup>T<sub>E</sub>X's appendix environment.

## 10.3 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.

---

**<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.

---

**<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.

## 10.4 Font change

---

**<em> – emphasis**

**Attributes:** None.

**Possible contents:** *(Text | inline element)\**

**Description:** Like `\emph{...}` in  $\text{\LaTeX}$ .

---

#### **<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  $\text{\LaTeX}$ 's `\text{??}` commands. "nm" switches to "normal" without changing the family. "vs" ("Versaliae") is intended for all-uppercase acronyms like "BASIC".

## **10.5 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  $\text{\LaTeX}$  command `\ref`. The referenced object, that is determined by `refid`, thus 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 immediatly before it, e. g.

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

This yields something like "table~2.1" for  $\text{\LaTeX}$ , 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  $\text{\LaTeX}$ 's `varioref` package which means that possibly 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  $\text{\LaTeX}$  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  $\text{\LaTeX}$  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:** Inserts a citation reference to `refid` (which can also be a space separated list of more than one citation). The attribute `kind` determines the kind of insertion. `kind="text"` would mean

Einstein et. al. (1921)

whereas `"paren"` makes

(Einstein et. al. 1921)

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

Einstein et. al. 1921

You know this probably from the `natbib` package, that’s behind all this. *Important:* So that this also works in HTML, the key entries of the `BIBTEX` file must look like `"text"`.

`"nocite"` includes nothing in the text, just in the references list at the end. By the way,

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

works like `\nocite{*}`  $\LaTeX$ . (You can't use `'*'` because `refid` is of type `NMTOKENS`, and such attributes mustn't contain `'*'`.)

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 et. al. (1921, `\emph{first}` chapter)"

The default value for `kind` is `"text"` except where the `<cite>` is directly surrounded by parentheses in the document. Then it's `"imparen"`.

---

### **`<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`.)

## **10.6 $\LaTeX$ -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 section 5 on page 16. *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  $\left[ \dots \right]$  in  $\text{\LaTeX}$ , but uses a somewhat different expression syntax, see section 5 on page 16.

---

**<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 formulas with e. g. unslanted chemical element symbols, see section 5 on page 16. E. g.

`<ch>AlxGa1-xAs</ch>`

yields “Al<sub>x</sub>Ga<sub>1-x</sub>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>`.

## 10.7 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, corollar, 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 ommitted 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"><subject>Streetmentioner's Corollary</subject>A
  short mathematical corollary. If it's corollary
  number 4, the next lemma will have number 5.</theorem>
```

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

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).

With the `layout` attribute, you can choose another style. Prefefined are the AMST<sub>E</sub>X styles `plain`, `definition`, and `remark`. If you want to add another one, you have to give its definition in a L<sup>A</sup>T<sub>E</sub>X package.

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

---

## **<proof> – mathematical proof**

### **Attributes:**

<code>xml:lang</code>	human language (inherited)
<code>id</code>	unique name (identifier)
<code>style/class</code>	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.

## 10.8 Miscellaneous

---

### **<url> – external reference**

#### **Attributes:**

name || the URL (required)

**Possible contents:** None.

**Description:** The name is printed with typewriter style and is linked with its own contents. Just as `\url{...}` would do in `hyperref` mode.

---

### **<hspace> – horizontal skip**

#### **Attributes:**

dim || width (required)

**Possible contents:** None.

**Description:** Makes a horizontal skip, as the `\hspace` macro of the same name.

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

should do the same as one `\quad`.

---

### **<relax>**

**Attributes:** None.

**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**

**Attributes:** None.

**Possible contents:** *Text*

**Description:** Inserts a physical quantity, see section 5 on page 16.

`<unit>3 m</unit>`

yields in  $\text{\LaTeX}$  “ $3\,\text{m}$ ”. 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&cdot;10^{-11} m^3 kg^{-1} s^2</unit>.`

(notice the spaces!) yields

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

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>` – $\text{\LaTeX}$ -Code

### Attributes:

<code>code</code>	$\text{\LaTeX}$ code (required)
<code>desperate</code>	"true" or "false" (default) – element for last phase of production?

**Possible contents:** Arbitrary.

**Description:** This is some sort of `\special` command: It inserts `code` if we’re producing  $\text{\LaTeX}$  output, and interprets the element’s contents else. It’s totally ignored if `desperate` is "true".

The idea behind `desperate` is that the trafo `tbook`  $\rightarrow$   $\text{\LaTeX}$  can be forced to interpret even `<latex>es` with `deperate="true"`, like here:

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

This is usually done in the last phase of production, maybe for having better page breaks. Another example is the “ $\text{\LaTeX}$ ” logo which can be achieved by

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

in the XML preamble and using it with “`&LaTeX;`”. On the other hand, this is a standard `tbook` entity anyway.

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

---

## `<wrap>` – inline wrapper

### Attributes:

<code>xml:lang</code>	human language (inherited)
<code>id</code>	unique name (identifier)
<code>style/class</code>	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 L<sup>A</sup>T<sub>E</sub>X's `\label{NicePosition}`.

## 10.9 Quoted and verbatim material

---

### **<quote> – displayed quotation**

**Attributes:**

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

**Possible contents:** *(Text | inline element)\**

**Description:** Inserts a displayed quotation.

---

### **<verb> – preformatted inline material**

**Attributes:** None.

**Possible contents:** *Text*

**Description:** Basically the same as L<sup>A</sup>T<sub>E</sub>X's `\verb`.

---

### **<verbatim> – preformatted displayed material**

**Attributes:**

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

**Possible contents:** *(Text | em | visual | ix | idx | indexsee)\**

**Description:** Corresponds to L<sup>A</sup>T<sub>E</sub>X's `verbatim` environment. In this context XML's `<![CDATA[ . . . ]>` is sometimes useful. Please note that – in contrast to L<sup>A</sup>T<sub>E</sub>X – some formatting elements are allowed. So you may print things in bold face, for example.

---

### **<verse> – lyrics**

**Attributes:**

<code>xml:lang</code>	human language (inherited)
<code>id</code>	unique name (identifier)
<code>style/class</code>	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.

---

**<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. <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>.

## 10.10 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.

---

**<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.

---

**<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.

---

#### **`<item>` – list item**

##### **Attributes:**

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

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

---

#### **`<term>` – description term**

##### **Attributes:**

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

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

**Description:** See `<description>`.

## **10.11 “Floats” and their contents**

---

#### **`<figure>` – floating figure**

##### **Attributes:**

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

**Possible contents:** `graphics, caption?`

**Description:** Encloses a figure and possibly a caption, that is then places 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, `tbook` decides whether the float spans both columns or not.

---

#### **`<table>` – floating table**

##### **Attributes:**

<code>xml:lang</code>	human language (inherited)
<code>id</code>	unique name (identifier)
<code>style/class</code>	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 L<sup>A</sup>T<sub>E</sub>X output, **tbook** decides whether the float spans both columns or not.

---

### <graphics>

#### Attributes:

file	filename <i>without extension</i> (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 L<sup>A</sup>T<sub>E</sub>X 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 my 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.

---

### <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!

---

### <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.
resize	"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:** centered on the same baseline.

**ccenter:** horizontally and vertically centered.

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.

resize 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 und 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:  $-1.5 \rightarrow -1,5$ ,  $-1.0 \rightarrow -1$ ,  $-0.5 \rightarrow -0,5$ ,  $0.0 \rightarrow 0$ ,  $0.5 \rightarrow 0,5$  and  $1.0 \rightarrow 1$ .

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.

## 10.12 Tables

---

### **<tabular> – table**

#### **Attributes:**

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

**Possible contents:** `tabhead?`, `tbody`

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

---

### **<tabhead> – headline of a table**

**Attributes:** None.

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

**Description:** Here every column gets a description.

---

**<tbody> – main matter of a table**

**Attributes:** None.

**Possible contents:** (hline | row | srow)\*

**Description:** Contains the actual data rows of a table.

---

**<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.

---

**<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.

---

**<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 L<sup>A</sup>T<sub>E</sub>X's booktabs package are always present and musn't be given explicitly.

---

#### **<cell> – table entry**

##### **Attributes:**

<code>colspan</code>	number of columns the entry occupies (default: "1")
<code>align</code>	"left", "center" or "right" – alignment (default: value from preamble)
<code>xml:lang</code>	human language (inherited)
<code>id</code>	unique name (identifier)
<code>style/class</code>	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 L<sup>A</sup>T<sub>E</sub>X, and `align` is clear, I think.

## **10.13 Elements of the frontmatter**

---

#### **<title> – title of the document**

##### **Attributes:**

<code>xml:lang</code>	human language (inherited)
<code>id</code>	unique name (identifier)
<code>style/class</code>	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.

---

#### **<author> – full name of one author**

##### **Attributes:**

<code>xml:lang</code>	human language (inherited)
<code>id</code>	unique name (identifier)
<code>style/class</code>	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 last-name, 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.

---

**<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.

---

**<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 (T<sub>E</sub>X) etc.

---

**<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 be printed unchanged as you've given it here.

---

**<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.

---

#### **`<year>` – of copyright**

**Attributes:**

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

**Possible contents:** *Text*

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

---

#### **`<city>`**

**Attributes:**

<code>xml:lang</code>		human language (inherited)
<code>id</code>		unique name (identifier)
<code>style/class</code>		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.

---

#### **`<legalnotice>`**

**Attributes:**

<code>xml:lang</code>		human language (inherited)
<code>id</code>		unique name (identifier)
<code>style/class</code>		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.

---

#### **`<abstract>` – summary of an article**

**Attributes:**

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

**Possible contents:** *p+*

## 10.14 Bibliography

---

**<references>** – list of cited material

**Attributes:**

bibfile	BIB <sub>T</sub> E <sub>X</sub> 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.

## 10.15 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.

---

**<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 command

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

looks in `tbook` like this:

```
<ix kind="emph">Sänger<ix2 sortkey="Twopac">2&nbsp;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.

---

### **`<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 additionally inserts its contents at the current text position.

---

### **`<indexsee>` – cross reference within index**

**Attributes:** None.

**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 always the one all the others are pointing to.

## 10.16 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"/>
```

---

### <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.

---

### <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  $\text{\LaTeX}$ . It begins the letter text.

---

**<closing> – end of letter**

**Attributes:**

<code>kind</code>	"above", "below" or "signature"
<code>xml:lang</code>	human language (inherited)
<code>id</code>	unique name (identifier)
<code>style/class</code>	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"/>
```