**HTML to \( \LaTeX \) transformation**
Frederik R. N. Schlupkothen

**Abstract**

\( (\LaTeX) \) was created as an authoring language that enables authors to keep typesetting quality standards while preparing their printed matter, assuming that the output context is known from the very beginning of the writing process. As a counter-concept, XML is focused on keeping output flexible, providing mechanisms to manage and control the logical structure of documents. Combining the strengths of both ecosystems has been discussed frequently in the past. This article aims to contribute to this discussion by introducing a mapping from HTML to \( \LaTeX \), the two most widespread document description languages in their respective fields of application.

**1 Introduction**

The Extensible Markup Language (XML) has become the native markup language for structured information in (distributed) document processing architectures. It provides a core syntax that has been adopted by many document description languages. A broad software ecosystem and further standards have emerged to realize and facilitate the processing of XML-based documents. Among them, the Extensible Stylesheet Language Transformations (XSLT) described in [8] offers a standardized mechanism to translate different XML-based languages into one another. This has made XML the language of choice for cross-media publishing workflows.

However, while the XML processing is fully covered by viable tools, the final document production of printed matter from XML may still be a problem. A potential solution is to integrate the \( \LaTeX \) typesetting engine (described in [10]) in XML-based processing chains. For this task, T\( \LaTeX \)XML, an XML representation of \( \LaTeX \) commands, was introduced in [12] and its “production proof” implementation discussed in [14]. As any other XML language, T\( \LaTeX \)XML documents can be produced via XSLT. So the last gap to fill in T\( \LaTeX \)XML-based workflows is to define XSL stylesheets that realize the transformation between specific XML and \( \LaTeX \) document description languages (see [14]).

Here we introduce a mapping between two document description languages that are well known in their respective fields of application: The HyperText Markup Language (HTML), as the transformation’s source language, is the core language of the World Wide Web and has a history that is closely tied to XML. It offers layout-oriented markup semantics primarily for textual content and is used amongst others to describe web pages (see [7]), electronic books (see [4]), and printed matter (see e.g. [9, 13]). \( \LaTeX \), as the transformation’s output, is the abstraction of \( \LaTeX \)’s typesetting commands to logical markup.

Figure 1 shows the XML-based production workflow that produces the Portable Document Format (PDF) from HTML by the use of the \( \LaTeX \) typesetting engine. The particular processing steps are described via an example of emphasized text:

1. The HTML source document describes the emphasized text “dolor” as: \(<em>dolor</em>\)
2. The XSLT processor queries the stylesheet “html2texml” for a matching transformation template. The matching template defines the transformation of the HTML’s “em” element to the corresponding \( \LaTeX \) command in its T\( \LaTeX \)XML representation:
   
   \[
   \begin{align*}
   &\text{xsl:template match="html:em"} \\
   &\text{\hspace{2em}tex:cmd name="emph";} \\
   &\text{\hspace{2em}\quad xsl:parm} \\
   &\text{\hspace{4em}xsl:apply-templates \} / \} \\
   &\text{\hspace{2em}\quad \}} \\
   &\text{\hspace{2em}} \\
   &\text{xsl:template}
   \end{align*}
   \]
3. The result of the XSLT transformation is the following T\( \LaTeX \)XML element:
   
   \[
   \begin{align*}
   &\text{cmd name="emph";} \\
   &\text{\quad parm\textit{dolor}\quad parm} \\
   &\text{\}} \\
   &\text{cmd}
   \end{align*}
   \]
4. The T\( \LaTeX \)XML processor converts the T\( \LaTeX \)XML element to a \( \LaTeX \) command: \texttt{\textbackslash emph\{dolor\}}
5. The \( \LaTeX \) processor typesets: \textit{dolor}

The prior example shows that defining the transformation between two languages needs insight into the differences in conceptual syntax and semantic coverage of source and destination language. While HTML is native to the XML syntax, T\( \LaTeX \)XML is reproducing the syntax logic inherited from \( (\LaTeX) \). While \( \LaTeX \) is native to printed matter, HTML was initially designed for electronic resources. Section 2 introduces the underlying concepts of the HTML and \( \LaTeX \) markup syntax. Section 3 introduces the mapping between HTML and \( \LaTeX \) markup semantics. Finally, section 4 shows a complete document with its corresponding representations in HTML and \( \LaTeX \).
2 Markup syntax

The following two subsections give an overview of the very basic HTML and LATEX syntax. They do not introduce the full syntax but focus on the aspects needed within this article. Full descriptions can be found in [3] for XML, the underlying markup language of HTML as described in [6], and in [10, 11] for LATEX.

2.1 Basic syntax of HTML

An HTML document consists of elements that are either empty or non-empty. The boundaries of a non-empty element is marked by a start-tag and an end-tag. Tag delimiters are the < and > characters. The element type is defined in the start-tag by its name. The end-tag repeats the element name preceded by a / character. The element’s content is enclosed between the start- and end-tag. The content consists of character data (i.e. text), subordinated child elements, or both. An element without content is called empty and is either described by a start-tag that is directly followed by its end-tag or by an empty-element-tag. An empty-element-tag has the same form as a start-tag but ends with a / character. The following example shows a non-empty ‘p’-element with mixed content:

```
<p>Lorem ipsum <em>dolor</em> sit amet. </p>
```

An element can possess attributes. Attributes are noted in the start-tag or empty-element-tag behind the element name. Attributes consist of a name-value-pair. The attribute-value is given between two ' or " characters and is assigned to its name by a preceding = character. The following example shows an empty ‘img’ element with a ‘src’ attribute of the value ‘uri’:

```
<img src="uri" />
```

There is exactly one root element that includes all the document’s content. The tag placement within the document follows the rules of mathematical brackets. The examples below show possible tag placements by means of ‘a’ and ‘b’ elements:

```
sequence       <a><b></a></b>
syntax error   <a><b></a></b>
```

2.2 Basic syntax of LATEX

A LATEX document consists of commands that describe either output characters (i.e. characters to typeset), special characters (e.g. the \ character for a non-breaking space), or control sequences. There are two types of control sequences: control words and control symbols. A control word starts with a \ character followed by its name that consists of one or more letters (i.e. lower- or uppercase letters ‘a’ to ‘z’) and is terminated by either a space or another non-letter. A control symbol starts with a \ character followed by one non-letter. A command can possess optional and required parameters that are set by arguments. Optional parameter arguments are noted after the command name between square brackets, and required parameter arguments between curly braces. The following example shows a ‘usepackage’-command with an optional parameter set to ‘utf8’ and a required parameter set to ‘inputenc’:

```
\usepackage[utf8]{inputenc}
```

Furthermore there are two special types of commands: environments and declarations. Environments are pairs of ‘begin’- and ‘end’-commands that enclose the environment’s content. The environment name is provided as the first required argument of the corresponding ‘begin’- and ‘end’-commands. The arguments of the environment are noted as further arguments of the ‘begin’-command. Declarations influence the behavior of following commands. The scope (i.e. range of effect) of most declarations is limited to its enclosing environment or group. The group delimiters are the { and } characters. The placement of group delimiters and environment commands follows the rules of mathematical brackets. The examples below show possible placements by example of a group and an ‘x’-environment:

```
\begin{x} \begin{x} \end{x} \end{x}
```

3 Markup correspondence

The following sections introduce a possible mapping between HTML elements and LATEX commands in the order of the HTML module descriptions in [1]. For an XSLT implementation transforming HTML to LATEX, the following mapping tables show the resulting LATEX commands for expository purposes.
3.1 Core Modules

The HTML Core Modules assemble the markup that is common to all HTML dialects that are derived from module-based HTML. This core markup for high level structures, basic text, hyperlinks, and lists of HTML documents and its corresponding \LaTeX{} commands are described in the following subsections.

3.1.1 Structure Module

The HTML Structure Module defines the high level markup of a document. The html-element is the document’s root containing the meta-information (head) and the actual content (body) of a document. \LaTeX{} follows a similar separation with its preamble and document-environment. Table 1 below shows the corresponding commands.

<table>
<thead>
<tr>
<th>HTML</th>
<th>\LaTeX{}</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;html&gt;...( ...)</td>
<td>\documentclass{report}...( )</td>
</tr>
<tr>
<td>&lt;head&gt;...( ...)</td>
<td></td>
</tr>
<tr>
<td>&lt;title&gt;...( ...)</td>
<td>\title{...( ...)}</td>
</tr>
<tr>
<td>&lt;body&gt;...( ...)</td>
<td>\begin{document}{...)}</td>
</tr>
</tbody>
</table>

Table 1: HTML to \LaTeX{} structure mapping

3.1.2 Text Module

The HTML Text Module defines the basic text markup to describe heading, block, and inline elements. Most of these elements have equivalent commands in \LaTeX{}, but not all. In these cases the ‘↦’ symbol indicates the default formatting in HTML where the Presentation Module described in section 3.2.1 might be used for an alternative, not corresponding semantically, mapping.

**Headings** The HTML Text Module defines six levels of headings (h1 to h6). \LaTeX{} offers a specific heading hierarchy that depends on the given document class. Table 2 below shows the corresponding heading commands for the report document class.

<table>
<thead>
<tr>
<th>HTML</th>
<th>\LaTeX{}</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;h1&gt;...( ...)</td>
<td>\chapter{...( ...)}</td>
</tr>
<tr>
<td>&lt;h2&gt;...( ...)</td>
<td>\section{...( ...)}</td>
</tr>
<tr>
<td>&lt;h3&gt;...( ...)</td>
<td>\subsection{...( ...)}</td>
</tr>
<tr>
<td>&lt;h4&gt;...( ...)</td>
<td>\subsubsection{...( ...)}</td>
</tr>
<tr>
<td>&lt;h5&gt;...( ...)</td>
<td>\paragraph{...( ...)}</td>
</tr>
<tr>
<td>&lt;h6&gt;...( ...)</td>
<td>\subparagraph{...( ...)}</td>
</tr>
</tbody>
</table>

Table 2: HTML to \LaTeX{} heading mapping

<table>
<thead>
<tr>
<th>HTML</th>
<th>\LaTeX{}</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;p&gt;...( ...)</td>
<td></td>
</tr>
<tr>
<td>&lt;address&gt;...( ...)</td>
<td>\emph{...( ...)}</td>
</tr>
<tr>
<td>&lt;blockquote&gt;...( ...)</td>
<td>\begin{quote}{...)}</td>
</tr>
<tr>
<td>&lt;div&gt;...( ...)</td>
<td>\begin{alltt}{...)}</td>
</tr>
</tbody>
</table>

Table 3: HTML to \LaTeX{} block mapping

**Blocks** The HTML Text Module defines elements to mark text groups as paragraphs (p), contact information (address), quotations (blockquote), generic groups (div), and preformatted text (pre). Table 3 shows the corresponding \LaTeX{} commands (using the \LaTeX{} core package alltt for preformatted text).

<table>
<thead>
<tr>
<th>HTML</th>
<th>\LaTeX{}</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;p&gt;...( ...)</td>
<td></td>
</tr>
<tr>
<td>&lt;address&gt;...( ...)</td>
<td>\emph{...( ...)}</td>
</tr>
<tr>
<td>&lt;blockquote&gt;...( ...)</td>
<td>\begin{quote}{...)}</td>
</tr>
<tr>
<td>&lt;div&gt;...( ...)</td>
<td>\begin{alltt}{...)}</td>
</tr>
</tbody>
</table>

Table 4: HTML to \LaTeX{} inline mapping

<table>
<thead>
<tr>
<th>HTML</th>
<th>\LaTeX{}</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;abbr&gt;...( ...)</td>
<td>\acrshort{...( ...)}</td>
</tr>
<tr>
<td>&lt;acronym&gt;...( ...)</td>
<td>\ac{...( ...)}</td>
</tr>
<tr>
<td>&lt;cite&gt;...( ...)</td>
<td>\cite{...( gen-id)}</td>
</tr>
<tr>
<td>&lt;q&gt;...( ...)</td>
<td>\enquote{...( ...)}</td>
</tr>
<tr>
<td>&lt;dfn&gt;...( ...)</td>
<td>\emph{...( ...)}</td>
</tr>
<tr>
<td>&lt;code&gt;...( ...)</td>
<td>\lstinline{...( ...)}</td>
</tr>
<tr>
<td>&lt;samp&gt;...( ...)</td>
<td>\teletype{...( ...)}</td>
</tr>
<tr>
<td>&lt;var&gt;...( ...)</td>
<td>\emph{...( ...)}</td>
</tr>
<tr>
<td>&lt;kbd&gt;...( ...)</td>
<td>\teletype{...( ...)}</td>
</tr>
<tr>
<td>&lt;em&gt;...( ...)</td>
<td>\textit{...( ...)}</td>
</tr>
<tr>
<td>&lt;strong&gt;...( ...)</td>
<td>\textbf{...( ...)}</td>
</tr>
<tr>
<td>&lt;span&gt;...( ...)</td>
<td></td>
</tr>
<tr>
<td>&lt;br&gt;</td>
<td>\newline</td>
</tr>
</tbody>
</table>

3.1.3 Hypertext Module

The HTML Hypertext Module defines markup to describe hyperlinks. They are described by source anchors (a) that reference to contents inside or outside of the document via Unified Resource Identifiers (URIs). Referenceable document fragments are marked by common ‘id’ attributes that can be applied to all elements. The use of traversable hyperlinks is an adequate solution in the context of electronic documents; its mapping to corresponding
\section*{3.1.4 List Module}

The HTML List Module defines markup to describe ordered (ol) and unordered (ul) lists as sequences of list items (li) and furthermore markup to describe definition lists (dl) that are composed of sequences of term (dt) and description (dd) pairs. Table 6 below shows corresponding \LaTeX{} commands.

\begin{table}[h]
\centering
\begin{tabular}{|l|l|}
\hline
HTML & \LaTeX{} \\
\hline
\texttt{<ol>\ldots} & \texttt{\begin{enumerate}\ldots} \\
\texttt{<ul>\ldots} & \texttt{\begin{itemize}\ldots} \\
\texttt{<li>\ldots} & \texttt{\item \ldots} \\
\texttt{<dt>\ldots} & \texttt{\begin{description}\ldots} \\
\texttt{<dd>\ldots} & \texttt{\item[]\ldots} \\
\hline
\end{tabular}
\end{table}

\subsection*{3.2 Text Extension Modules}

The HTML Text Extension Modules assemble additional text markup to control text rendering, maintenance, and direction for HTML documents. These and the corresponding \LaTeX{} commands are described in the following subsections.

\subsection*{3.2.1 Presentation Module}

The HTML Presentation Module defines markup to control the text rendering. It provides elements to render text in/as bold (b) and italic (i) style, typewriter (tt), super- (sup) or subscripted (sub), larger (big) or smaller font (small). Additionally the module provides an element to render horizontal rules (hr). \LaTeX{} offers corresponding commands with the exception of ‘textsuperscript’ that relies on the \texttt{subscript} package. The \texttt{relsize} package offers commands to realize relative font sizes (as intended by the ‘big’ and ‘small’ elements in HTML). Table 7 shows a possible mapping.

\subsection*{3.2.2 Edit Module}

The HTML Edit Module defines editing-related markup. It provides elements to mark content as deleted (del) or inserted (ins). The \texttt{changes} package offers semantically corresponding \LaTeX{} commands as shown in Table 8 below. However, if \LaTeX{} is used as final output format only, a more stable solution might be to simply output contents of ‘ins’-elements, but not those of ‘del’-elements.

\begin{table}[h]
\centering
\begin{tabular}{|l|l|}
\hline
HTML & \LaTeX{} \\
\hline
\texttt{<del>\ldots} & \texttt{\deleted{\ldots}} \\
\texttt{<ins>\ldots} & \texttt{\added{\ldots}} \\
\hline
\end{tabular}
\end{table}

\subsection*{3.2.3 Bi-directional Text Module}

The HTML Bi-directional Text Module defines markup to declare text direction changes. It provides an attribute to control the direction of text (dir) that can be applied to all elements including a special element (bdo) to override the current text direction. The \texttt{bidi} package offers corresponding \LaTeX{} commands. Table 9 below shows the corresponding commands for inline text. However, the \texttt{bidi} package defines a set of new environments which replace common \LaTeX{} commands (e.g. lists and footnotes) which makes the general mapping between elements and commands more complex. Furthermore the combination with other common packages (e.g. \texttt{hyperref} or \texttt{longtable}) remains problematic. So a more stable solution might be to omit bi-directional text controls during the transformation process and to apply such changes manually in the \LaTeX{} document.

\begin{table}[h]
\centering
\begin{tabular}{|l|l|}
\hline
HTML & \LaTeX{} \\
\hline
\texttt{<bdo dir="ltr">\ldots} & \texttt{\LR{\ldots}} \\
\texttt{<bdo dir="rtl">\ldots} & \texttt{\RL{\ldots}} \\
\hline
\end{tabular}
\end{table}

\subsection*{3.3 Forms Modules}

The HTML Forms Modules define markup to describe interactive forms that can define, organize, and receive (textual) input and selections. The \texttt{hyperref}

Frederik R. N. Schlupkothen
3.4 Table Modules

The HTML Table Modules define markup to describe tables (table) by organizing their data (td) and header (th) cells in rows (tr). These rows can be grouped into table headers (thead), footers (tfoot), and bodies (tbody). Column-based markup is realized by standoff elements (col and colgroup). A table caption (caption) can provide a short description of the table contents.

\( \text{LaTeX} \) table definitions differ in two essential aspects from HTML: (i) The total number of table columns has to be given explicitly to a \text{LaTeX} table environment. This is not necessary in HTML but calculated continuously by the rendering engine at processing time. (ii) \text{LaTeX} table cells that span several rows (by means of the multirow package) cover the adjacent cells in the following rows; therefore empty cells need to be inserted in the following rows. This is not necessary in HTML but the rendering engine automatically shifts the cells of the following rows according to the reading direction.

Hence for the transformation of HTML tables to \text{LaTeX} this information (total number of table columns and position of additional empty cells) need to be precalculated. Therefore the transformation process has to include parts of the HTML table processing model described in \cite{hyperref}. This model describes an HTML table as a set of cells that are positioned on a two-dimensional grid of slots. The algorithm shown in Figure 2 calculates the cell positioning and illustrates how the additional empty cells are inserted; hence the total number of table columns is given by the maximum \( x \)-coordinate within the final grid. Table 11 shows the mapping of HTML table elements to corresponding \text{LaTeX} commands by means of the longtable package.

\begin{table}[h]
\centering
\begin{tabular}{|l|l|}
\hline
HTML & \text{LaTeX} \\
\hline
<form>(...) & \begin{Form}(...) \\
<input /> & \TextField{} \\
type="password" & \TextField[password]{\langle label\rangle} \\
type="checkbox" & \TextField{} \\
type="button" & \PushButton{\langle label\rangle} \\
type="radio" & \ChoiceMenu{\langle label\rangle}= \}
\hline
type="submit" & \Submit{\langle label\rangle} \\
type="reset" & \TextField{} \\
type="file" & \TextField[textfield]{\langle label\rangle} \\
type="hidden" & \TextField[hidden]{\langle label\rangle} \\
type="image" & \TextField[submitcoordinates]{\langle img\rangle} \\
<select>(...) & \ChoiceMenu{\langle label\rangle} \langle options\rangle \\
<option>(...) & \langle...\rangle \\
textarea{...} & \TextField{\langle multiline\rangle} \\
button{...} & \Submit{\langle...\rangle} \\
type="button" & \PushButton{\langle...\rangle} \\
type="reset" & \TextField{} \\
<textfield> & \TextField{\langle...\rangle} \\
<legend> & \langle...\rangle \\
<optgroup> & \langle...\rangle \\
\hline
\end{tabular}
\caption{HTML to \text{LaTeX} forms mapping}
\end{table}

\begin{figure}[h]
\begin{verbatim}
table = \{row_i \mid row_i = \{cell_{ij}\}\}
grid = \{slot_i \mid slot_i = (x, y)\}

procedure TABLE(table)
grid \leftarrow \{\emptyset\}

for all row_i \mid i = 1..n do
  y \leftarrow i
  x \leftarrow 1

for all cell_{ij} \mid j = 1..m do
  while grid \ni (x, y) do
    empty CELL(x, y)
    x \leftarrow x + 1
  end while
  for y_{cell} \ni 0..rowspan(cell_{ij}) - 1 do
    for x_{cell} \ni 0..colspan(cell_{ij}) - 1 do
      grid \leftarrow grid \cup \{(x + x_{cell}, y + y_{cell})\}
    end for
  end for
  x \leftarrow x + colspan(cell_{ij})
end for
end procedure
\end{verbatim}
\caption{HTML table cell positioning algorithm}
\end{figure}

The HTML Image Module defines markup to embed external images. The graphicx package offers a corresponding \text{LaTeX} command as shown in Table 12.

3.6 Further Modules

The HTML specification describes further modules that define markup to realize dynamic and interactive document content, mechanisms to control layout, and deprecated markup for backwards compatibility with legacy HTML. Due to the focus of this article on the transfer of the logical structure of HTML documents
Table 11: HTML to \LaTeX{} table mapping

<table>
<thead>
<tr>
<th>HTML</th>
<th>\LaTeX{}</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;caption&gt;...</td>
<td>\caption{...}</td>
</tr>
<tr>
<td>&lt;table&gt;...</td>
<td>\begin{longtable}{</td>
</tr>
<tr>
<td>&lt;td&gt;...</td>
<td>(...)&amp;</td>
</tr>
<tr>
<td>&lt;th&gt;...</td>
<td>\bf{}(...) &amp;</td>
</tr>
<tr>
<td>colspan=&quot;span&quot;</td>
<td>\multicolumn{\colspan*}{{...}</td>
</tr>
<tr>
<td>rowspan=&quot;span&quot;</td>
<td>\multicolumn{\rowspan*}{{...}</td>
</tr>
<tr>
<td>&lt;tr&gt;...</td>
<td>{...}|</td>
</tr>
<tr>
<td>&lt;col/&gt;</td>
<td></td>
</tr>
<tr>
<td>&lt;colgroup&gt;...</td>
<td></td>
</tr>
<tr>
<td>&lt;tbody&gt;...</td>
<td></td>
</tr>
<tr>
<td>&lt;thead&gt;...</td>
<td></td>
</tr>
<tr>
<td>&lt;tfoot&gt;...</td>
<td></td>
</tr>
</tbody>
</table>

Table 12: HTML to \LaTeX{} image mapping

<table>
<thead>
<tr>
<th>HTML</th>
<th>\LaTeX{}</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;img src=&quot;(uri)/&quot;/&gt;</td>
<td>\includegraphics{(uri)}</td>
</tr>
</tbody>
</table>

to \LaTeX{}, the mapping of these specialized modules is not described in detail. However, in specific use cases the support of these modules might be desired. The following hints might serve as a starting point to implement a transformation of these modules’ features to \LaTeX{}.

The HTML Applet, Object, Scripting, and Intrinsic Events modules define markup that introduces scripting facilities to manipulate dynamically the document content. At present this is notably realized through the JavaScript programming language, which is partially integrated with \LaTeX{} by means of the insdljs package.

The HTML Client- and Server-side Image Map modules define markup for interactive and hyperlinked images. This functionality can be potentially realized in \LaTeX{} by means of the Ti\TeX{} package.

The HTML Frames and Iframe modules define markup to insert one document into another. This can be realized in \LaTeX{} with the \texttt{\input} and/or \texttt{\include} commands.

The HTML Style Sheet and Style Attribute modules define markup to integrate layout definitions realized through Cascading Style Sheets (CSS). CSS has its own syntax and description logic—its transformation to \LaTeX{} is a topic all its own, which has been outlined e.g. in [15].

4 An example

Figure 3 shows an example page taken from a bird guide. On the next page, Figure 4 shows a possible coding of this page using HTML, and Figure 5 its corresponding representation in \LaTeX{}.

Gannet

Birds of the open ocean, Gannets breed on small islands off the NW coast of Europe. They move away from land after nesting to winter at sea. The young migrate south as far as W Africa. Gannets feed on fish by plunge-diving from 25m. They nest in large, noisy colonies. The nest is a pile of seaweed. A single egg is incubated for 44 days. The young bird is fed by both parents and flies after 90 days.

![Gannet](image)

Figure 3: An example document, derived from [5]

5 Conclusion

While HTML is increasingly becoming the common document description language for different output media (web, print, e-books, . . . ), the problem of creating well-typeset documents from HTML is not yet fully solved within the XML ecosystem. The article at hand has introduced a mapping from HTML elements to corresponding \LaTeX{} commands, in order to use the Ti\TeX{} typesetting engine for this task.

With the multitude of existing \LaTeX{} extensions released as packages, almost any HTML description can be ported to \LaTeX{} and typeset according to its original logic. Unfortunately, the use of \LaTeX{} packages often comes with a catch: while many HTML structures can be used recursively (e.g. nested lists or tables), \LaTeX{} packages tend to override existing commands giving them a new meaning (e.g. the newline command is redefined in table environments to end a row). These context-dependent syntax-changes can make a mapping potentially error-prone for deep document structures.

Frederik R. N. Schlupkothen
Birds of the open ocean, Gannets breed on small islands off the coast of Europe. They move away from land after nesting to winter at sea. The young migrate south as far as Africa. Gannets feed on fish by plunge-diving from 25 meters. They nest in large, noisy colonies. The nest is a pile of seaweed. A single egg is incubated for 44 days. The young bird is fed by both parents and flies after 90 days.

![Gannet](gannet.jpg)

<table>
<thead>
<tr>
<th>Size</th>
<th>Larger than any gull</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adult</td>
<td>White, black wing-tips, yellow nape</td>
</tr>
<tr>
<td>Juvenile</td>
<td>Grey, gradually becoming white over 5 years</td>
</tr>
<tr>
<td>Bill</td>
<td>Dagger-like</td>
</tr>
<tr>
<td>In flight</td>
<td>Cigar-shaped with long, narrow, black-tipped wings</td>
</tr>
<tr>
<td>Voice</td>
<td>Usually silent, growling &quot;urr&quot; when nesting</td>
</tr>
<tr>
<td>Lookalikes</td>
<td>Skuas, Gulls and Terns</td>
</tr>
</tbody>
</table>

**Figure 4**: HTML source, describing the document shown in Figure 3

```latex
\documentclass{report}
% preamble ...
\begin{document}
\chapter{Gannet}

Birds of the open ocean, Gannets breed on small islands off the coast of Europe. They move away from land after nesting to winter at sea. The young migrate south as far as Africa. Gannets feed on fish by plunge-diving from 25 meters. They nest in large, noisy colonies. The nest is a pile of seaweed. A single egg is incubated for 44 days. The young bird is fed by both parents and flies after 90 days.

\includegraphics{gannet.jpg}

\begin{longtable}{ll}
\toprule
\bf{}Size & Larger than any gull \\
\bf{}Adult & White, black wing-tips, yellow nape \\
\bf{}Juvenile & Grey, gradually becoming white over 5 years \\
\bf{}Bill & Dagger-like \\
\bf{}In flight & Cigar-shaped with long, narrow, black-tipped wings \\
\bf{}Voice & Usually silent, growling "urr" when nesting \\
\bf{}Lookalikes & Skuas, Gulls and Terns \\
\bottomrule
\end{longtable}

\end{document}
```

**Figure 5**: \LaTeX{} output, exported from HTML shown in Figure 4

HTML to \LaTeX{} transformation
The mappings introduced in this article have been developed in the context of an XSLT implementation within a \TeXXML-based workflow, but do not rely on it and can be implemented through other approaches as well. However, the principle of \TeXXML, to provide a processor that transforms specific \TeX commands from a generic XML representation to the \TeX format, realizes a separation between the task of format transformation and the task of defining appropriate mappings. This facilitates the definition and adaption of markup correspondences as has e.g. been done by extending the HTML mapping with a third party stylesheet that defines the transformation from MathML to \BiTeX.

Acknowledgment
The author would like to thank Prof. Dr. Karl-Heinrich Schmidt for giving valuable advice and Gilles Bilow for integrating this document processing workflow into our daily tasks for testing purposes.

References

Frederik R. N. Schlupkothen
University of Wuppertal
Rainer-Gruenter-Str. 21
D-42119 Wuppertal
Germany
schlupko (at) uni-wuppertal dot de