A bit of HTML and a bit of ConTeXt

HTML constructs translated to ConTeXt

Abstract

Described is a module for the typesetting of a subset of HTML operators. These can be used to build data sets in XML with HTML as formatting elements and have them typeset in ConTeXt. Other features are the inclusion of predefined content and provision for language localized words and expressions.

Keywords

ConTeXt, HTML, XML, include, vocabulary

1 Introduction

Let us start with a tiny example, just to get the idea. For the sake of exposition the content of this example is enclosed as if it were HTML source. That is not necessary though, any root node can contain these elements. For it to work one has to load the module with \use-module[hvdm-xml] first. It in turn depends on modules [hvdm-ctx] (a few general macros) and [hvdm-lua] (some Lua coded functions). By the way, note the mandatory XML-notation for the characters < and > and of course the same will apply to the ampersand & as these are XML-related (call it) features. Also note that the example code below can be rendered in a browser unchanged.

Example: <body>

roman italic bold

Below follows a description of the API of this module, here and there illustrated with an example. Many of the nodes are standard to HTML but there are some extras, primarily because the author needed them for his work. The idea behind the development of the module was to stay close to HTML and use its vocabulary where possible.

2 Document Structure

These are nodes for structuring your XML document.

- <html>
  The well known enclosure of all HTML.
- <head>
  The header part of the document. Its content is ignored because it is not supposed to play a role in the typesetting process.
- <body>
  The body part of the document. The content of this node can contain all the typeset material.
- <h1> through <h6>
  Typeset headers in diminishing size according to the digit behind the 'h'. The alignment can be left, center, middle or right, the color of the heading as well as its style can be specified. The possibilities in size are <h1> ... <h6>.

Example: <root>

heading text

</root>
Following are nodes for paragraphs and linebreaks.

- `<br/>
  Break the line.
- `<p height="" color="" align="" style=""/>
  Encloses a paragraph, although `<p>` does equally well as a simple paragraph break. Extra white can be placed above the paragraph with the height attribute. The content can be colored, aligned and given a specific style, all with attributes.
- `<div/>
  Just a division as in HTML.

3 \TeX directly

These are nodes for those parts of the document where one has to resort to using \TeX directly. One such area is the typesetting of math, although it is possible to use MathML here. However, I find that a tedious business and for quick results am using \TeX's math directly. MathML translation can come later, when things have to be tidied up.

- `<tex/>
  Contains pure \TeX or Con\TeXt.
- `<m>
  Encloses the node content in pair of `$`s.
- `<M>
  Encloses the node content in pair of `$$`s.
- `<nohyphens>
  Stop hyphenation and put the content inside a `\begingroup-\endgroup` pair, as might be expected.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<root>
  Before –
  <font family="euler" size="big">
    Fontswitch
  </font>
  – After
</root>
```

### Before – Fontswitch – After

```xml
<?xml version="1.0" encoding="UTF-8"?>
<root>
  <smaller step="2pt">
    smaller
  </smaller>
  <normal>
    <big step="2pt">
      bigger
    </big>
  </normal>
</root>
```

4 Appearance

These are nodes for making font, style and color changes.

- `<font family="" size="" style=""/>

```xml
<?xml version="1.0" encoding="UTF-8"?>
<root>
  <r/>
</root>
```

The common HTML style nodes are present.
Typeset content in \tt.
- `<u space="yes/no">` Typeset content underlined, the space attributed governs the breaking of the underlining at word boundaries.
- `<i>` Typeset content in \it.
- `<b>` Typeset content in \bf.
- `<em>` Typeset content in \em.
- `<sl>` Typeset content in \sl.
- `<tt>` Typeset as in the corresponding HTML.
- `<code>` Typeset as in the corresponding HTML.

Special operations on text.
- `<sub>` Typeset content in subscript.
- `<sup>` Typeset content in superscript.
- `<lohi>` Typeset content both in subscript and in superscript. The node should contain a `<sub>` and a `<sup>` node, in any order.
- `<quote>` and `<q>` These enclose their content in single and double quotes, respectively.

5 Space and Alignment

These are nodes for structuring the typesetting with horizontal and vertical space.
- `<spacer size="" type="">` By default the kind of spacing depends on whether the typesetting is currently in horizontal or in vertical mode. But its kind can be explicitly specified by setting the attribute `type` to either `horizontal` or `vertical`. A third type `fill` is evaluated to `\hfill` or `\vfill` respectively. By default the `size` values are `1em` and `\normalbaselineskip` for the horizontal and vertical displacements. The size may also be given as a percentage, which is applied to `\textwidth` or `\textheight`.
- `<center>` Typeset centered.
- `<left>` Typeset aligned to the left.
- `<right>` Typeset aligned to the right.
- `<narrow left="" right="" middle="" option="">` Narrowed paragraph, the option attribute determines the narrowing as in the command `\startnarrower`, middle is default. The other attributes determine the value of the narrowing and evaluate to calling command `\setupnarrower`.
- `<blockquote>` The HTML blockquote behaviour.

6 Lists

These are nodes for enumerations, mimicking those in HTML.
- `<li>` List element.
- `<ol sym="" start="" columns="" option="">` Ordered list with attribute `sym` specifying the kind of numbering: `n`, `a`, `A`, `r`, `R`, `g`, `G` for numbers, letters, roman numerals or Greek letters. The `start` attribute determines the number of the first item, by default 1 as expected. The `columns` attribute specifies the number of columns (1 by default). Attribute `option` functions as sort of catch all, because its content is transferred to the first `[ ]`-argument of the typesetting `\startitemize` macro.
- `<ul sym="" columns="" option="">`
Unordered list with the symbol having a great number of variants, switched by a number from 1 to 14 and ranging from bullets to triangles and lozenges (provided the font has them).

- `<dl compact="yes/no">`
  As in HTML, encloses `<dt>` and `<dd>` elements.
- `<dt>`
  As in HTML.
- `<dd>`
  As in HTML.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<root>
<ul sym="3" columns="2">
  <li>first item</li>
  <li>second item</li>
  <li>third item</li>
  <li>fourth item</li>
</ul>
</root>
```

7 Tables

These are nodes for tables, mimicking those in HTML and translated to XML-table macros of ConTeXt.

- `<table foregroundstyle="" offset="" columndistance="" spaceinbetween="" location="" option=""/>
  The setup attributes implemented here are
  foregroundstyle corresponding to HTML’s style,
  offset corresponding to cellpadding, columndistance to cellspacing, spaceinbetween to rowspacing and
  location for placement top, left or middle. The catch
  all option attribute can be used to set frameoffset, backgroundoffset, textwidth, textheight,
  leftmarginistance, rightmargdistance.
- `<thead>`
  As in HTML.
- `<tfoot>`
  As in HTML.
- `<tbody>`
  As in HTML.
- `<tr>`
  As in HTML.
- `<td>`
  As in HTML.
- `<th>`
  As in HTML.

8 Images and graphics

These are nodes for typesetting images and pictures.

- `<framed many-attributes>`
  Frame the contents of the node. There are a great many of attributes here, corresponding with
  the most important ones from \setupframed. To
  name them without further explanation: height, width, offset, color, bgcolor, frame, framecolor, framecorner, framerateiness, strut, align, valign, corner, radius, bgcorner, bgradius, option. The last one option can be anything and its
  value is passed on to the underlying setup call. Frame parts may be specified with HTML-like attributes for the frame attribute and has values
  on, t, above, vsides is topframe=on
  on, b, below, vsides is bottomframe=on
  on, l, lft, ldsides is leftframe=on
  on, r, rhs, ldsides is rightframe=on
- `<img>`
  Place an image file through \externalfigure. Separate attributes are height, width, scale, frame, corner, radius, rotation, option. These are inserted in the
  second []-argument to \externalfigure. The dimensions might be given as a percentage (for example
  50% or 50p), which is taken as a fraction of \text-theight and \textwidth respectively. The rotation
  attribute specifies a rotation of the figure given in degrees.
- `<mpgraphic name="" [file="" buffer="" parameters=""]`
  Execute MetaPost on an MPgraphic definition and
  place this in the document. The node must contain a
  \startuseMPgraphic \stopuseMPgraphic pair.
  Other sources for the definition of the graphic may
  be specified with the file and buffer attribute.
  The attribute parameters is used to transfer variable
  values via the \MPvar macro to the MetaPost code.
  Note that the name on the \startuseMPgraphic must
  be the same as the one on the <mpgraphic>.

Example where the graphic is defined in a separate
ConTeXt buffer:

```
\startbuffer[graphic]
\startuseMPgraphic{example}{color}
  pickup pencircle scaled 1mm;
  draw unitsquare scaled 1cm
  withcolor \MPvar{color};
\stopuseMPgraphic
\stopbuffer

% Called with:
<mpgraphic buffer="graphic" name="example"
  parameters="color=darkblue"/>
```
Example where the graphic is defined in the \texttt{<mpgraphic>} node itself.

\begin{verbatim}
\texttt{<mpgraphic name="example" parameters="color=darkgreen">}
\startuseMPgraphic{example}{color}
pickup pencircle scaled 1mm;
draw unitcircle scaled 1cm withcolor \MPvar{color};
\stopuseMPgraphic
\texttt{</mpgraphic>}
\end{verbatim}

Example where the graphic is defined in a file.

\begin{verbatim}
\texttt{<mpgraphic name="example" file="example-file" parameters="color=orange"/>}
\end{verbatim}

9 Include and definition nodes

Instead of repeatedly copying the same code over and over, one can define these common parts separately (\texttt{includes.xml} in the example below) in a file. This file must contain a root node in which the definitions of the common code parts are put. The particular name of this node \textit{is not significant}; it is merely a container for the \texttt{<define>} nodes inside. In the example below \texttt{<includes>} was chosen.

Each of the definitions contained in the file should be given a name whereby it can be called up for insertion. Loading the file with definitions is done with the \texttt{<include>} node carrying the path of the file to read. A missing file attribute is an error.

\begin{verbatim}
<?xml version="1.0" encoding="UTF-8"?>
<includes>
  <define name="mypart">
    .. code of mypart ..
  </define>
</includes>
</?xml version="1.0" encoding="UTF-8"?>
<root>
  <include file="includes.xml"/>
</root>
\end{verbatim}

A \texttt{<define>} node carrying content is a \textit{definition} node. If more than one \texttt{<include>} file is loaded, then later inclusions have priority above the earlier ones. The first match of a definition breaks off the search.

A \texttt{<define name="name"/>} node not carrying content implies \textit{retrieval} of the definition. Thus the presence or absence of the node’s content determines its role. In the following example code the definition \textit{mypart} is substituted for \texttt{<define name="mypart"/>}.

\begin{verbatim}
<include>
  <define name="mypart">
    .. code of mypart ..
  </define>
</include>
</text><define name="mypart"/></text>
\end{verbatim}

The \texttt{<define>} can have a \texttt{type} attribute if it is a definition node. In that case some special action depending on its value is taken when called up. The preprogrammed actions are:

1. \texttt{type="image"} When the \texttt{<define>’s} content resolves to the location of a file, this is put into the document as an \texttt{\externalfigure}.
2. \texttt{type="mpgraphic"} The \texttt{<define>} must resolve to code for a graphic.

For example a definition can be

\begin{verbatim}
<define type="mpgraphic" name="example" parameters="color=blue,variant=0">
\startuseMPgraphic{example}{color,variant}
pickup pencircle scaled 1mm;
if \MPvar{variant} = 0:
draw unitcircle scaled 1cm
else:
draw unitsquare scaled 1cm rotated 45
fi
withcolor \MPvar{color};
\stopuseMPgraphic
</define>
\end{verbatim}

On the call the default color blue keeps its value, but...
the variant is changed to 1. Note that the names on the <define> and the definition of the graphic must be the same. We called this definition with

<define name="example" parameters="variant=1"/>

The above image was defined and placed with the code below. The <define>-node contains the name of the image file.

<define type="image" name="example">
  sauron.jpg
</define>

...<define name="example" parameters="width=75p"/>

Beware. The inclusion of a <define>-node is not the same as macro substitution in a programming language like for instance C. Formula in MathML are good candidates for a <define>.

10 Vocabulary

It is not uncommon to program in English for all terms that can appear in the typeset document. However, it would be nice if instead these could be customized and typeset in the target language selected with the ConTeXt macro \\language[]. The <vocabulary>-node can help to enlighten this task. With it one defines for a chosen series of words the equivalent in each of several languages. The next example defines the English word file, the Dutch bestand and the German Datei as equivalents. Note the use of the two letter language designators.

<vocabulary>
  <word>
    <language name="en">file</language>
    <language name="nl">bestand</language>
    <language name="de">Datei</language>
  </word>
</vocabulary>

One of the languages will function as the reference language in which the program specifies the terms (English by default). The macro calls \translate{file} and \Translate{file} are typeset as bestand and Bestand for either the ‘nl’ or the ‘de’ current language. The same is accomplished with the nodes <word> and <Word>.

Further customization is the possibility to specify another language as the reference language than the default English. Specify the reference language as its two letter value as a vocabulary attribute <vocabulary reference="">. Likewise the language into which the words are translated can be enforced through the use attribute <vocabulary use="">. Be aware of the fact that mixing vocabularies within different reference languages may have unexpected results.

The vocabulary can be placed in a file or in a buffer whose name must be given as attribute:

<vocabulary file="" buffer=""/>

11 Convenience Nodes

Not HTML but probably useful now and then. A save-restore, current time and date macros are provided for:

- <currendate/>
  Typesets the current date in European style.
- <currenttime/>
  Typesets the current time in a 24-hour clock.
- <store name="">
  Store content of the node under name.
- <restore name="">
  Call up content stored under name.

<?xml version="1.0" encoding="UTF-8"?>
<root>
  store then <store name="test">
    stored text
  </store>
  restore: <restore name="test"/>
  <div/>
  current date: <currendate/>
  <div/>
  current time: <currenttime/>
</root>

store then restore: stored text
current date: 08-04-2016
current time: 20:58

Hans van der Meer
H.vanderMeer@uva.nl