Document Markup for the Web

Document markup is the process of adding codes to a document to identify the structure of a document and to specify the format in which its fragments are to appear. We will discuss two conflicting aspects — structure and appearance — in document markup. As the Internet imposes special constraints imposed on markup formats, we will reflect its influence.

In the past few years the XML format has established itself as a general basis for markup languages. As OMDoc and all mathematical markup schemes discussed here are XML applications (instances of the XML framework), we will go more into the technical details to supply the technical prerequisites for understanding the specification. We will briefly mention XML validation and transformation tools, if the material reviewed in this section is not enough, we refer the reader to [Har01].

1.1 Structure vs. Appearance in Markup

Text processors and desktop publishing systems (think for example of Microsoft Word) are software systems aiming to produce “ink-on-paper” or “pixel-on-screen” representations of documents. They are very well-suited to execute typographic conventions for the appearance of documents. Their internal markup scheme mainly defines presentation traits like character position, font choice and characteristics, or page breaks. We will speak of presentation markup for such markup schemes. They are perfectly sufficient for producing high-quality presentations on paper or on screen, but for instance it does not support document reuse (in other contexts or across the development cycle of a text). The problem is that these approaches concentrate on the form and not the function of text elements. Think e.g. of the notorious section renumbering problems in early (WYSIWYG) text processors. Here, the text form of a numbered section heading was used to express

1 “What you see is what you get”; in the context of markup languages this means that the document markup codes are hidden from the user, who is presented with a presentation form of the text even during authoring.
the function of identifying the position of the respective section in a sequence of sections (and maybe in a larger structure like a chapter).

This perceived weakness has lead to markup schemes that concentrate more on function than on form. We will call them content markup to distinguish them from presentation markup schemes, and discuss \TeX/\LaTeX{} \cite{Knu84, Lam94} as an example.

\TeX{} is a typesetting markup language that uses explicit markup codes (strings beginning with a backslash) in a document, for instance, the markup \verb|$\sqrt{\sin x}$| stands for the mathematical expression \(\sqrt{\sin x}\) in \TeX{}. To determine from this functional specification the visual form (e.g. the character placement and font information), we need a document formatting engine. This program will transform the document that contains the content markup (the “source” document) into a presentation markup scheme that specifies the appearance (the “target” document) like DVI \cite{Knu84}, POSTSCRIPT \cite{Rei87}, or PDF \cite{PDF06} that can directly be presented on paper or on screen. This two-stage approach allows the author to mark up the function of a text fragment and leave the conversion of this markup into presentation information to the formatter. The specific form of translation is either hard-wired into the formatter, or given externally in style files or style sheets.

\LaTeX{} \cite{Lam94} is a comprehensive set of style files for the \TeX{} formatter, the heading for a section with the title “The Joy of \TeX{}” would be marked up as

\begin{verbatim}
\section{The Joy of \TeX{}}
\end{verbatim}

This piece of markup specifies the function of the text element: The title of the section should be “The Joy of \TeX{}”, which (if needed e.g. in the table of contents) can be abbreviated as “\TeX{}”, the glyph “\TeX{}” is inserted into the index, where the word tex would have been, and the section number can be referred to using the label sec:TeX. Note that renumbering is not a problem in this approach, since the actual numbers are only inferred by the formatter at run-time. This, together with the ability to simply change style file for a different context, yields much more manageable and reusable documents, and has led to a wide adoption of the function-based approach. So that even word-processors like MS Word now include functional elements. Pure presentation markup schemes like DVI or POSTSCRIPT are normally only used for document delivery. On the other hand, many form-oriented markup schemes allow to “fine-tune” documents by directly controlling presentation. For instance, \LaTeX{} allows to specify traits such as font size information, or using

\begin{verbatim}
\textbf{proof}: \ldots \hfill \Box
\end{verbatim}

to indicate the extent of a proof (the formatter only needs to “copy” them to the target format). The general experience in such mixed markup schemes is that presentation markup is more easily specified, but that content markup will enhance maintainability and reusability. This has led to a culture of style file development (specifying typographical and structural conventions), which now gives us a wealth of style options to choose from in \LaTeX{}}.