

---

# Paper Template for WWW2004

F R A Hopgood

---

## Contents

- [1. Introduction](#)
  - [2. Template](#)
  - [3. Printing from XHTML](#)
  - [4. The Tutorial](#)
- 

## 1. Introduction

At our last meeting I agreed to develop an XHTML Template for use at WWW2004.

The parts of the action were:

- FRAH to prepare XHTML template will help from Ivan Herman and Janet Daly
- Wendy Hall's group to confirm that the metadata could be extracted simply from this format
- To develop guidelines for papers with mathematics
- To provide an interim version for use at WWW2003
- Wendy Hall's Group to define the set of keywords they would like
- FRAH to provide a tutorial on formatting issues for Budapest

## 2. Template

The file [template.htm](#) is a first stab at a document to be sent to authors describing the format to be used for WWW2004. The document itself uses the format. The associated stylesheet, [iw3c2.css](#), approximates to the format used by ACM for their printed version. An early version of this was used as the submission format for WWW2003. I believe this template should be easy to extract the metadata that Wendy's group needs but have not had a positive response back from them. It has not been passed to ACM for comment. Ivan believed it would be most appropriate to discuss it with ACM immediately after WWW2003. We would then have had another year's experience with ACM as publisher and would have some feedback from using the interim template.

The major change has been a much stronger emphasis on using MathML for mathematics. There has been a dramatic change in the support for MathML by Mozilla, IE and Netscape since October 2002. W3C has provided an XSLT transformation to hide the IE non-standard behaviour markup so that it is now feasible to recommend that mathematics should be provided in the paper submissions as MathML islands in the XHTML. Most of the existing systems for displaying mathematics have converters to MathML and there is a good site that provides information and support.

## 3. Printing from XHTML

To achieve the layout that ACM requires (2 column with good support for page layout variations) probably means that the easiest route to generating the published version of the Proceedings is to use an XSLT transformation on the document to generate XSL-FO output to drive the printer.

The file [acm.xml](#) is the simple example document and [acm.xsl](#) gives the flavour of the XSLT transformation needed to transform this into XSL-FO. The file [acm.fo](#) is the input document transformed into XSL-FO. I assume ACM have more of an XSL-FO expert than I am. Most of the large publishing houses are familiar with this kind of transformation. My stylesheet is not an industry strength stylesheet but it gives the flavour of what is required. the file [acm.pdf](#) shows the kind of output that you can produce. I have deliberately used colour to differentiate parts of the document just to help in understanding which bits of output come from where. I am using a demo version of Antenna House's XSL-FO processor. Apache's FOP is probably a better alternative for an organization like ACM, where server side production probably makes sense, or one of the commercial systems such as XEP.

If we decide to go this route, it will be necessary to work out where ACM is in terms of modern XML-based printing. Almost certainly either mathematics or images will cause problems as they do for any other production process. What is clear now is that any problems are solvable. Inevitably it will not be perfect and I do not believe the current process is either.

## 4. The Tutorial

### 4.1 XHTML

Figure 1 shows the current state of HTML. XHTML became the **standard** version of HTML over 3 years ago so asking people to mark up documents in XHTML is not being innovative. It is what people should be doing. In Europe, Government departments are required by law to have information available on the web in Strict XHTML. XHTML1.0 comes in three flavours, Strict, Transitional and Frameset. Transitional allows you to use the styling attributes that were in HTML4.0 while Strict does not. Given that we are trying to make the Proceedings available on different media, it is most sensible to insist that papers are submitted in XHTML Strict Format. Anything else and there is a lot of work removing the styling from individual elements.

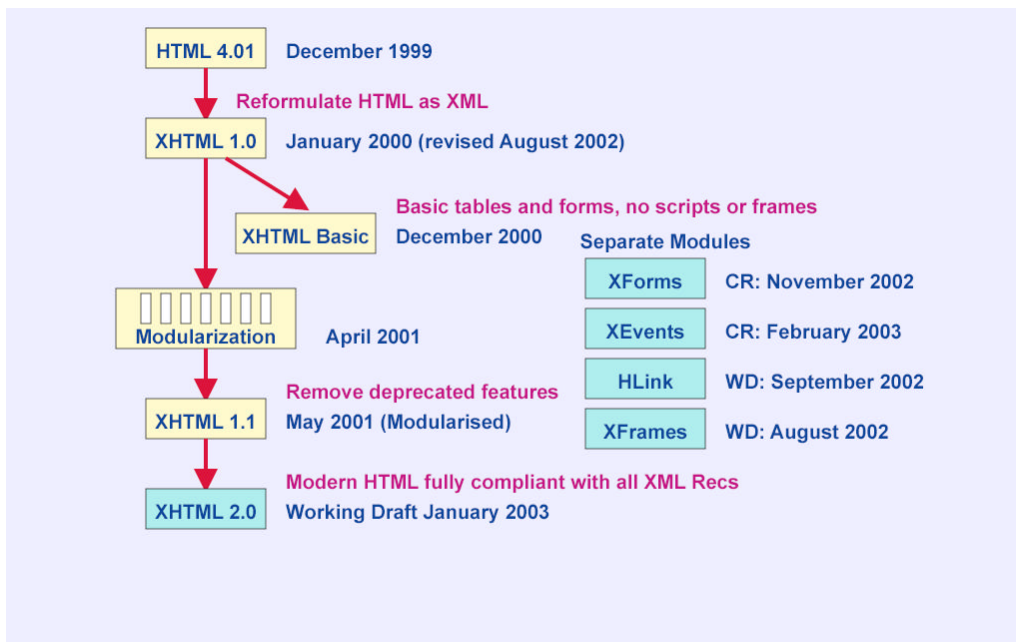


Figure 1. XHTML Roadmap

In terms of markup, there is not much difference between XHTML 1.0 and XHTML 1.1 while it is too early to think about using XHTML 2.0.

### 4.2 XSLT and XSL-FO

Presentation of XHTML documents on the Web can either be done using CSS or XSLT. If you want to display the document in the sequential order that it is marked up, it probably makes sense to use CSS. All the modern browsers support a large subset of CSS2 and some features of CSS3. Pretty well the whole of CSS1 is supported. CSS provides good control of styling and layout of paragraphs and inline elements but only provides rudimentary control of pages (and some print features are not supported by all browsers). So if you want good control of pagination, widows and orphans, support for indexing and contents, hyphenation, and all the things you can do in PDF, then you need to use XSL-FO as the output format with XSLT transforming the XHTML document into XSL-FO.

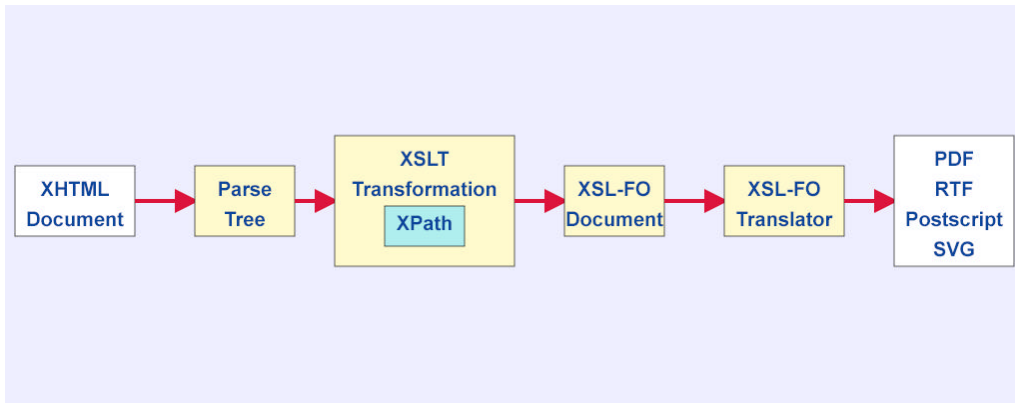


Figure 2. XSLT and XSL-FO

Figure 2 illustrates the process. The XHTML document is presented to an XSLT transformation engine which first builds a parse tree of the document and then transforms it using XSLT. Parts of the tree to be processed are identified using XPath. The result is another XML document. It could be XHTML but for printing, the output document is an XSL-FO document. Pretty well any XSLT transformation can be used for this phase. As most XSL-FO engines also have an XSLT transformer in them, it is likely that you will use the one in the XSL Formatting Engine. The Formatting Engine takes the XSL-FO document and can usually generate output in a variety of formats, predominantly print ones although you can produce aural or display versions. The Formatting Engines try to provide all the output formats they think people require. The major one that they nearly all output is PDF.

The XSLT document is predominantly a set of templates that say if you see a tree node like this one, transform it into this. In our particular case, the XHTML document is in three parts, each with a **div** element enclosing it. The first part is the title/author/metadata part. The second is the document itself and the third part is the set of references. I chose this as Wendy needs to get hold of the first and third but does not need to look at the second. An XSLT transformation could just ignore the document itself and pull out Wendy's metadata.

XSL-FO documents are pretty unreadable and verbose. This tends not to matter as people seldom read them. As long as the engine generates the correct output all you tend to need to look at are the XSLT transformation and the printed result.

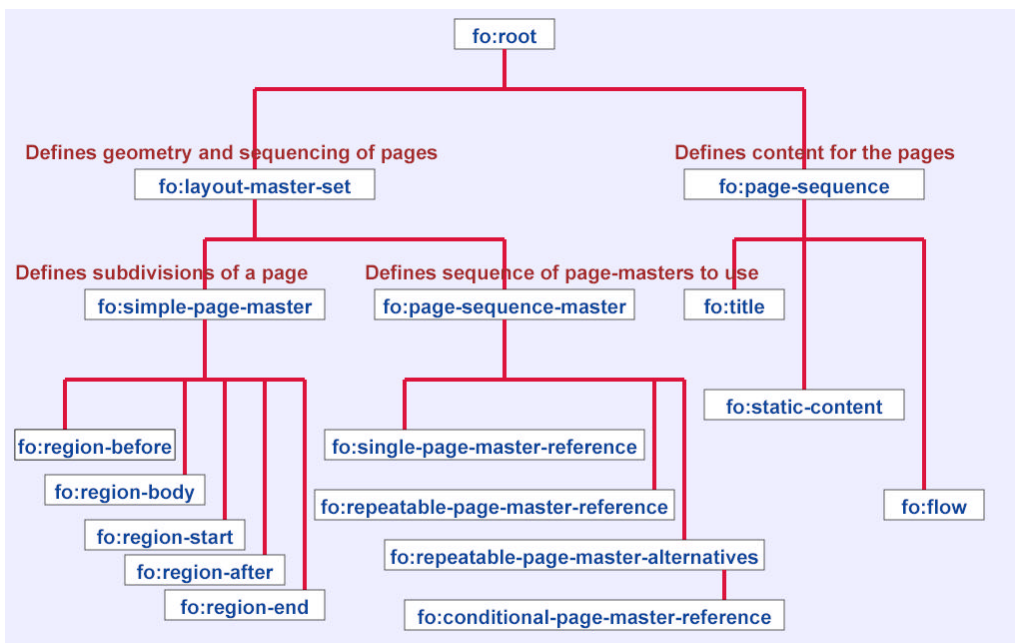


Figure 3. XSL-FO Document

Figure 3 gives an overview of what an XSL-FO document look likes. The left hand side defines a set of formats for how you want the physical sheet of paper to look. This says how big the sheet of paper is, what margins you want etc. This is equivalent to the Page Size section in the ACM Template Document. You may have several of these or just one depending on how complicated the document us. In our case as ACM does not differentiate between left and right pages, we can probably get away with two, the first page and the rest.

The middle part defines which page formats you use when. In our case we would say you use one template for the first page and another for the rest.

The XSL-FO model is that you then have a bucket of content that you pour into the pages. You can pour the content into the main central area of the page and into areas above, below, left and right. As ACM adds most of the page numbering etc afterwards, we only have to bother with the middle. ACM might add what it needs to do to produce the whole volume. So our XSLT transformation generates lots and lots of elements inside the **fo:flow** element. For each **h1**, **p** **table** element, you have to say what the margins, padding, font, justification etc are for it. Luckily you don't have to write all this. All the author has to do is to have an XSLT template for each of the things you want to output. So in the main body of the document you will have a template for, say, the **p** element which says use 9pt Times Roman with no margin above and a small margin below and the text justified.

The **fo:static-content** element is used for adding page titles, page numbers etc.