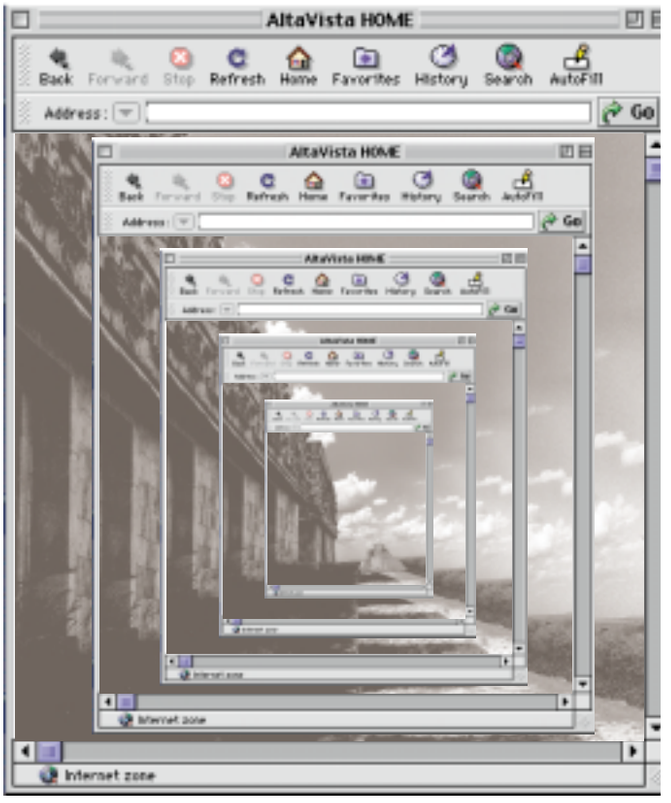
Multipurpose Web Publishing

Using HTML, XML, and CSS

The World Wide Web Consortium devised these document-markup and style-sheet languages in the interests of Web device independence, content reuse, and network-friendly encoding.

Håkon Wium Lie and Janne Saarela

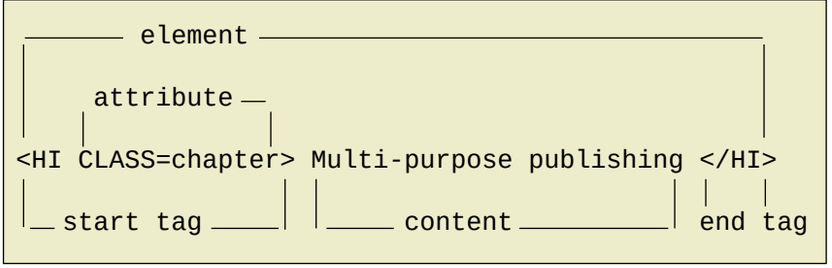


Since its conception in the early 1990s, the Web has become a critical component in the strategic thinking of content providers around the world. But targeting the Web as the delivery vehicle for content poses several questions, including: How should the publishing process change to take advantage of the Web? and How should content be represented to support device independence, searchability, and efficient network throughput?

The protocols and data formats in use on the Web constitute a framework within which applications and services can be built. Emerging Web standards can be used to implement *multipurpose publishing*, where the same content is presented on a range of Web devices.[1] We discuss three specifications in some detail—the HyperText Markup Language (HTML), the eXtensible Markup Language (XML), and Cascading Style Sheets (CSS). All three can help content providers face some of the most important challenges of delivering content on the Web:

[1]We use the term "Web device" to denote any hardware or software through which a user accesses Web content.

Figure 1. Structure of SGML markup.



Device independence. As the Web grows, the number and range of Web devices increases. Screens of varying sizes, printers, and speech synthesizers are among the devices on which content will be presented. Will it be necessary to encode content for each new device, or are device-independent formats possible?

Content reuse. To preserve investment, it's important that content be reusable in many different ways. Search engines must be able to find and index pages, archives must be maintainable, and the content must be in a format accessible by future Web devices.

Network-friendly encodings. For Web users, having to wait for pages to download is among the chief frustrations. One of the main reasons for slow downloads is the use of images to encode textual content in order to preserve style. Network-friendly encodings have to found while remaining true to the publisher's style.

The World Wide Web Consortium (W3C) is a coordinating body working with its more than 300 member organizations in developing the Web's underlying technical specifications. Several of the W3C's recent Recommendations extend the function of the Web in ways that will be significant for content providers, including those concerning HTML, XML, and CSS.

Structured Document Markup Languages

Computer encodings of documents have long concentrated on preserving the "final form presentation," such as a nicely laid-out paper document. Structured document formats take a different approach; rather than preserving the final form presentation, they encode the document's logical structure. Among the reasons for doing so is the preservation of device independence, document searchability, and information re-use in general.

The Standard Generalized Markup Language (SGML), which became an ISO standard in 1986, pioneered the concept of structured documents [5]. The philosophy behind SGML is to define a general meta-language that can be used to build application-specific languages to encode structured documents. A language specification—in SGML called a Document Type Definition (DTD)—defines the elements, element containment, and element attributes used to mark-up a document instance (see Figure 1). Several document instances may be valid SGML documents and conform to the same DTD.

Traditionally, elements in SGML encode structure, rather than presentation. For example, the headline of a document is marked as being a headline, rather than specifying a particular font size. This adds one level of indirection; so, in order to find, say, the font size of the headline, a style sheet has to be consulted. The style sheet describes the presentation of documents.

Although a number of vendors offer SGML-compliant products, SGML is a complex technology requiring significant investment by the content provider. In the past few years, work on structured documents has centered on simplifying SGML; two of these efforts are HTML and XML.

HTML. HTML has its roots at the European Laboratory for High-Energy Physics (CERN) in Geneva, Switzerland, where the Web project began in 1990. At that time, HTML served physicists who needed to collaborate by sharing scientific articles over the Internet. Although for most of us, the content of these articles is difficult to comprehend, their document structure is quite simple. This structure is reflected in the small set of general elements in HTML, including headings, paragraphs, lists, and anchors for hyperlinks. The semantics in HTML is sparse but known by millions of Web devices around the world.

HTML was formally specified as an SGML DTD in 1992, giving the HTML specification a context in which further expansion was possible, though it also conflicted with sentiments in the early Web community. Because SGML is a complex technology, implementing a full SGML parser was beyond the interests of early Web application developers. This resulted in browsers that accepted non-valid documents; as a result, even today few documents on the Web are valid according to the HTML specification. Moreover, although HTML came from the structured documents community, it was influenced by "presentational" document formats, including Postscript. HTML still contains such elements as "B" (for bold)

and "I" (for italics) that encode document presentation rather than structure. This breaks with the SGML principle of separating structure from presentation.

Today, the set of HTML elements has stabilized at around 80. New elements are being added slowly through the W3C working group on HTML that publishes revisions of the HTML specification. HTML 4.0 is the latest version and contains several noteworthy features for content providers [12]. First, HTML 4.0 deprecates the use of a large set of elements that mainly encode presentation; their function is better served by style sheets.

HTML 4.0 also adds a CLASS attribute on all elements. By using this attribute, elements can be subclassed into categories of choice—in effect creating new elements. The CLASS attribute can hold information that would otherwise be lost when converting a document to HTML, and a style sheet can act on the value of the CLASS attribute (see Figure 2).

Having a designated W3C working group in charge of HTML development has been a stabilizing factor for the language. No vendor can single-handedly add new elements to HTML, and the document format remains nonproprietary. Moreover, the semantics of the various elements is well known. For example, all browsers and search engines know that the "H1" element indicates a first-level headline. Thus, HTML has achieved a unique position as a device-independent, ubiquitous document format.

The downside of the committee approach is that communities in need of additional markup (beyond subclassing existing HTML elements) cannot easily build on HTML. For example, mathematicians may want to encode formulae inside HTML documents, but HTML lacks special elements for mathematics. For such applications, XML comes to the rescue.

XML. In light of the limited repertoire of HTML elements, content providers cannot easily encode semantics into their documents. An initiative to regain the advantages of SGML on the Web began in 1996 when a W3C working group was formed to identify a subset of SGML suitable for the Web. Later known as XML, the initiative has gathered support from both the SGML and the Web communities.

XML includes SGML's ability to define new elements. For content providers, this means XML can encode semantics more gracefully than HTML. In addition, XML removes the burden of having to validate documents against a DTD; XML documents may refer to a DTD but are not required to do so. Instead, a document can claim to be well-formed by following some simple syntactical rules.

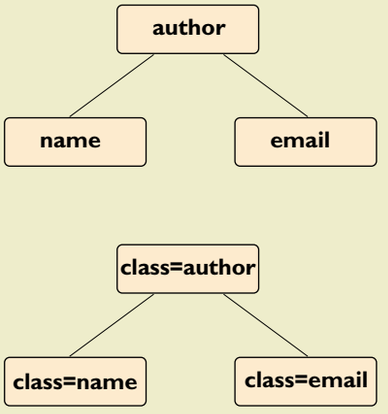
Figure 2. Example A shows element containment within an SGML document instance. Example B shows how similar containment heirarchy can be achieed by complementing HTML semantics with the CLASS attribute.

```
Example A:

    <author>
      <name>Janne Saarela</name>
      <email>jsaarela@w3.org</email>
    </author>

Example B:

    <DIV CLASS=author>
      <DIV CLASS=name>Janne Saarela</DIV>
      <DIV CLASS=email>jsaarela@w3.org</DIV>
    </DIV>
```



The XML specification became a W3C Recommendation in February 1998, and its first uses have appeared [1]. For example, two new Web data formats—the Synchronized Multimedia Integration Language (SMIL) [6] and the Resource Description Framework (RDF)—are written in XML.

RDF is a metadata infrastructure format allowing content providers to encode metadata, or information about information, in machine-understandable form. RDF unifies the field of metadata by allowing authors to use assertions from different schemas, such as the Dublin Core (DC) [3] and the Platform for Internet Content Selection (PICS) [11], in a single classification entry.

The example in Figure 3 shows how DC elements are qualified with the DC prefix within an

Figure 3. An RDF document mixing assertions from multiple schemas.

```
<?xml version="1.0"?>
> <RDF:RDF xmlns:RDF="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
>          xmlns:DC="http://purl.org/dc/elements/1.0/">
>   <RDF:Description about="http://www.w3.org/TR/NOTE-multipurpose">
>     <DC:title>Multipurpose publishing using HTML, XML, and
> CSS</DC:title>
>     <DC:language>en</DC:language>
>     <DC:creator>
>       <RDF:Bag ID="authors">
>         <RDF:li>Hakon Lie</RDF:li>
>         <RDF:li>Janne Saarela</RDF:li>
>       </RDF:Bag>
>     </DC:creator>
>   </RDF:Description>
> </RDF:RDF>
```

ß

RDF/XML document. The metadata entry in the Figure gives information on the electronic counterpart of the article.

The knowledge representation in metadata is a double-edged sword; keeping the encoding simple makes metadata easy to read but with less "expressivity," whereas a more complex encoding allows for more expressive semantics. RDF builds on a common syntax—XML—and a simple data model based on nodes and arcs. Specialized applications can help in the authoring of these seemingly complex descriptions.

Figure 4. A simple XML fragment with associated CSS.

```
Markup:
   <author>
    <name>Janne Saarela</name>
    <email>Jsaarela@w3.org</email>
  </author>

Style sheet:
  author { font: 12pt Times }
  name { font-weight: bold }
  email { font-style: italic }
```

SMIL and RDF both use XML to describe structure (using tags) but have little or no content (such as text). Strictly speaking, these formats are not "markup" languages but represent a trend, in that most XML on the Web is used for data, not for documents.

Style Sheets

The notion of style sheets is complementary to structured documents; documents contain content and structure, and style sheets describe how documents are to be presented. This separation is a requirement for device-independent documents (all device-specific information is left to the style sheet) and simplifies document management, since a style sheet can describe many documents.

For example, if an XML document uses element names, such as "author," "name," and "email" (see Figure 4), there is no hint as to how to present the content on, say, A4 paper.

CSS. Work on CSS began at CERN in 1994 with the goal of developing a style-sheet language for the Web that would fulfill author requests for stylistic control beyond HTML. In 1996, CSS1 (the first level of CSS) became a W3C Recommendation [8]; in 1997, support for CSS1 was added to major browsers, including Netscape Navigator 4 and Microsoft Internet Explorer 4, as well as to various authoring tools.

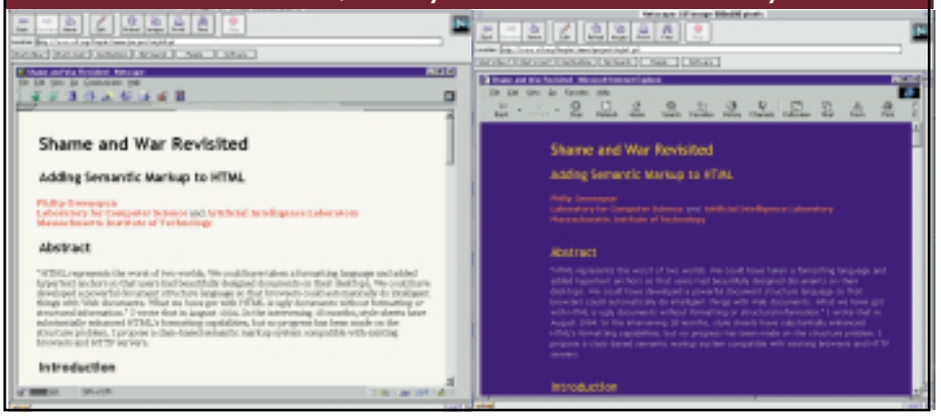
CSS uses declarative rules to attach style to elements. A simple rule might say that all P elements of class "warning" are to be displayed in red text on a white background:

```
P.warning {
  color: red;
  background: white;
}
```

CSS1 supports screen-based formatting, including fonts, colors, and layout (see Figure 5). Before style sheets, Web authors had to make pictures of text to convey colors and fonts. This has resulted in a Web in which most of the network bandwidth is used not for text but for pictures of text. Therefore, style

sheets have the potential for significantly improving network performance, as concluded by a recent study of how new Web technologies affect network performance [10], stating, "To our surprise, style sheets promise to be the biggest possibility of major network bandwidth improvements, whether deployed with HTTP/1.0 or HTTP/1.1, by significantly reducing the need for in-lined images to provide graphic elements, and the resulting network traffic."

Figure 5. The same document with two different CSSs. The document's underlying HTML source is identical; the only difference is the link to the style sheet.



Using style sheets instead of images also improves Web accessibility. A speech synthesizer can read HTML-encoded text to a blind user; the text can also be presented through a braille tactile feedback device. Images, on the other hand, deny nonvisual access.

The next level of CSS—CSS2, which became a W3C Recommendation in May 1998 [9]—strengthens Web accessibility by adding the concept of media-specific style sheets. For example, a style sheet can describe an aural rendering of a document, as in:

```
@media speech {
  BODY { voice-family: female }
  H1 { volume: loud }
}
```

This style sheet applies to all Web devices supporting speech output. Such media-specific style sheets enable designers to carefully describe presentations for groups of devices while the underlying documents remain device-independent.

Handheld Web devices also require special attention from style sheets because of their small display surface. For example, there may be no room for images, and only a shortened version of the document should be presented. The following style sheet turns off the display of images and normal paragraphs, so only paragraphs of class "ingress" are shown:

```
@media handheld {
  IMG { display: none }
  P { display: none }
  P.ingress { display: block }
}
```

A CSS style sheet is typically processed in the Web device itself. However, to save bandwidth for mobile handheld devices, it may be worthwhile to process the style sheet in a stationary proxy server. In the "ingress" example above, the style sheet turns off the display of images, so the proxy server can therefore withhold images from the mobile device. This way, valuable bandwidth is saved, and the Web's perceived performance is increased.

Taking Style Further

The Extensible Stylesheet Language (XSL), currently being defined by a W3C working group, takes the concept of style sheets a step further by being able to transform document structure. For example, an XSL sheet can automatically generate a table of contents by extracting all chapter titles from a document.

Using XSL to transform XML data into structured documents, such as HTML, will play an important part in multipurpose publishing for years to come.

But can style be taken too far? Among the first additions to HTML after it escaped from CERN were forms that let users interact with pages by filling in text fields and pressing buttons. Later, the introduction of scripts (such as JavaScript, now being standardized as ECMAScript) and Java applets enables applications to be distributed over the Web.

Many Web pages mix declarative data (such as HTML, XML, and CSS) with executable programs (such as scripts and applets). Content providers are often motivated to use programs to achieve special presentational effects (such as an animated headline or a pop-up menu). When aiming for multipurpose publishing, it's important to carefully consider the costs and benefits before relying on scripts and

applets to present information. Costs include:

Accessibility. Content embedded in a program is hidden from Web search engines, and it's difficult or even impossible to convert that content to other formats.

Maintainability. In 20 years, will machines be able to decode HTML files? Probably. Will machines be able to run current scripts and applets? Maybe. Declarative data is generally easier to maintain and lives longer than programs.

Device independence. Many scripts assume a graphical Web device and will not work on, say, a text-only browser.

As development of style sheets progresses, we expect that the most popular presentational effects achieved through programming will find their way into declarative style rules. For example, CSS2 includes functionality for highlighting an element when a mouse moves over it; up to now, such highlighting has been possible only through scripts.

In 1997, the W3C initiated an activity—called Document Object Model (DOM)—to describe the interface between programs and documents. Its goal is to define a language-independent application programming interface (API) that applications can use to access and modify the structure, content, and style of HTML and XML documents.

Presented on a Variety of Web Devices

Structured documents with style sheets allow the same document to be presented on a variety of Web devices. Indeed, the goal of multipurpose publishing is to need only one source document flexible enough for use in different environments. However, it may sometimes be necessary to translate the document from one representation format to another before publishing on the Web. But there are problems in managing the content in a different representation from the one actually served.

Our central claim is that HTML, together with style sheets, should be rich enough to serve as a master document format for many publishers. However, outside of traditional documents, other data formats written in XML can also capture semantics.

Capturing semantics for future applications. The terms "down-translation" and "up-translation" are often used in discussions of how to translate documents from one format to another. Down-translation refers to a process whereby the resulting document has less semantically significant markup available than the original source document. Up-translation refers to a reversed process whereby the source document can

be in any format, and specialized rules are used to remove presentation-oriented, often-proprietary markup. The goal is a higher-level representation with abstract markup elements suitable for a document description that is platform- and device-independent.

Up-translation can be viewed as a preparatory process for multipurpose publishing, leveraging the value of the information content so far that new applications can be implemented. The actual down-translation is often a straightforward process fine-tuned with in-house rules to perform the actual translation to other document formats.

The down-translation process also has to cope with the real-life requirements of the publishing schedule. For example, this process may take place once a year in the context of an encyclopedia or 50 times a second for an interactive online service.

Leveraging existing information with up-translation and learning to use new authoring tools or existing tools in new ways will represent an investment justified by the long-term value of the information. Several scenarios illustrate the sort of applications that might be possible with more semantic markup and metadata:

Structured queries. Locating documents through URLs or search engines is standard practice on the Web today. Structured queries would extend the range of search options by allowing queries for, say, a given author or location.

Alternate user interface metaphors. Web devices typically arrange visited documents in a linear list as they are traversed. An alternative user interface could create a virtual landscape of traversed links whereby, say, the location and political bias of authors are visualized.

Intelligent agents. Users place an offer for some merchandise, then have their agent find the best vendor on the Web. Users indicate interest in buying an item, and their agents search for and identify a maximum price and negotiation strategy. Then, without user intervention, these agents use the item description to find, say, all potential resellers and commits a financial transaction to inform the user of a shipment arriving tomorrow at 7:00 A.M.

These scenarios demonstrate there is room for improvement in the way information is represented and reused on the Web. An important catalyst for new Web applications will be the incorporation of additional semantics in machine-understandable form.

Recommendations

Simple HTML documents, augmented with style sheets, preserve device independence and accessibility while improving network performance. For anyone publishing documents on the Web, the ubiquitous HTML is likely to be the document format of choice for years to come.

Today, users should want more from the information they access. Those authoring HTML can enhance their content by using the full semantics of HTML and adding style sheets. Those authoring content in other formats before putting it on the Web should ensure that translating it to HTML preserves the original semantics. This requires additional effort during authoring but pays off as new Web applications become possible. XML allows content providers to encode highly structured data and should be given careful consideration when designing new Web applications.

In general, such declarative data formats as HTML, XML, and CSS are recommended over scripts and applets for stylistic effects in multipurpose publishing. Declarative data, which is easily converted to other formats, is more likely to be device-independent and tends to live longer than programs.

The Web is generous enough to accommodate any content we place there. We should therefore ensure that our content meets the Web's high standards. ■

REFERENCES

1. Bray, T., Paoli, J., and Sperberg-McQueen, C. *Extensible Markup Language (XML) 1.0 Specification.* W3C; see www.w3.org/TR/.
2. Coombs, J. Renear, A., and DeRose, S. Markup systems and the future of scholarly text processing. *Commun. ACM 30,* 11 (Nov. 1987).
3. Dublin Core Metadata Element Set; see purl.oclc.org/metadata/dublin_core/.
4. ECMA-262. *ECMAScript* (a general-purpose, cross-platform programming language). June 1997; see www.ecma.ch/stand/ecma-262.htm.
5. Goldfarb, C. *The SGML Handbook.* Oxford University Press, New York, 1990.
6. Hoschka, P. *Synchronized Multimedia Integration Language.* W3C; see www.w3.org/TR/.
7. Lie, H. and Bos, B. *The Cascading Style Sheets—Designing for the Web.* Addison-Wesley Longman, Harlow, England, 1997.
8. Lie. H. and Bos, B. *Cascading Style Sheets, Level 1.* W3C; see www.w3.org/TR/.
9. Lie. H., Bos, B, Lilley, C., and Jacobs, I. *Cascading Style Sheets, Level 2.* W3C; see www.w3.org/TR/.
10. Nielsen, H., Gettys, J., Baird-Smith, A., Prud'hommeaux, E., Lie, H., and Lilley, C. Network performance effects of HTTP/1.1, CSS1, and PNG. In *Proceedings of ACM SIGCOMM'97* (Cannes, France, 1997).
11. PICS. *Platform for Internet Content Selection 1.1 Specifications.* W3C; see www.w3.org/TR/.
12. Raggett, D., Le Hors, A., and Jacobs, I. *HTML 4.0 Specification.* W3C; see www.w3.org/TR/.

HÅKON WIUM LIE (howcome@opera.com) is chief technology officer of Opera Software in Oslo, Norway; before joining Opera, he was the style sheet activity lead at the W3C, where he proposed the CSS concept in 1994.

JANNE SAARELA (js@pro-solutions.com) is the managing director of Pro Solutions, Ltd. in Helsinki, Finland; from 1997 to 1999, he was a visiting scientist at the W3C French office at INRIA, Sophia-Antipolis, France.