Taming the Web with XML

by William J. "Bill" McCalpin, EDPP, CDIA, MIT, LIT

What is XML? Here's what some industry publications say about it: "XML will automate the Web," "XML Stakes Out Web Future Right through HTML's Heart," and "You need to start thinking about XML because a year from now you'll undoubtedly be using it a lot." Curious? Good. But just what do you need to know?

First, XML is eXtensible Markup Language. It is an instance of SGML—one of the family of architectures which comprises the SGML family. OK, so what's SGML? SGML is the Standard Generalized Markup Language (ISO 8879), the international standard for defining descriptions of the structure and content of different types of electronic documents. In other words, SGML is not a language itself but a way of defining a language so that it can accurately communicate information in a document— as well as communicate information about the information.

Industries and enterprises come together and set standards for common definitions within the SGML architecture. Since this means that each industry creates its own set of definitions, SGML is neither tied to nor dependent on one particular environment.

The Difference Between SGML And Print Data

Over the past 20 years, there have been many presentation datastreams. Line data—Xerox's Metacode, IBM's AFP, Adobe's PostScript and PDF, Hewlett-Packard's PCL—all use data with the same basic goal: presentation. When we print or display data in one of these formats, we specify presentation information: an X and Y location, a font, an orientation, and some text or data.

The foundation for SGML is quite different. In an SGML language, each piece of data is tagged with a description of the data's purpose. SGML makes it possible to not only present data itself but also the meaning of the data (the author's content) at the same time.

An SGML document is the synthesis of three things:

- A Document Type Definition (DTD)

- A Stylesheet
- Tagged Data

The DTD

The DTD describes the types of tags allowed in a document as well as the order in which they may appear. The DTD validates the names of the tags as well as the order of the tags which appear in a document. This makes it possible for all parties in a large project to guarantee that the data passed between them will always be mutually understandable. It is a table which can be used to syntax-check the document. Over the years, a variety of groups have agreed on hundreds of DTDs.

This sample DTD describes which tags (e.g., <page>) are allowed. The DTD states that a document consists of a heading (<head>) and a body (<body>). The DTD states that the body has zero or more pages, and each page contains a mixture of paragraphs (p), examples (ex), and unordered lists (ul).

Simple sample of a DTD:

```
<!element document (head, body)>
<!element head (#PCDATA)>
<!element body (page)>
<!element page (p | ex | ul)*>
<!element p (#PCDATA)>
<!element ex (#PCDATA)>
<!element ul (#PCDATA)>
```

But tags aren't limited to grammatical constructs like headings or paragraphs. The DTD could define tags like <Part-Number> or <Exchange-Rate>. Each piece of data in a datastream could be explicitly tagged with its purpose, as we will see below.

The Stylesheet

The stylesheet describes how the corresponding tag is formatted. For example, Microsoft's Word for Windows uses styles to format parts of a document. You might define a normal paragraph as Arial, normal, US English, flush left, line spacing single, widow / orphan control. But you could also create a style called check-number and give it the same formatting. In other words, formatting can be totally divorced from author's content.

The XML FAQ (Frequently Asked Questions) maintained by Web Consortium's XML Special Interest Group (see www.w3c.org) says this about formatting: "A new Extensible Style Language (XSL) is being proposed for use specifically with XML. This uses XML syntax (a stylesheet is actually an XML file) and combines formatting features from both DSSSL (the SGML standard) and CSS (HTML) and has already attracted support from several major vendors."

Tagged Data

The tagged data is your document, in which every piece of data is tagged, using tags defined in the DTD and the stylesheet.

Sample of tagged data:

```
<document><head>This is a sample
document using tagged data</head>
<body><page>
<p>This is text in a paragraph</p>
<ex>This is an example of some-
thing</ex>
</page></body></document>
```

Why Isn't HTML Good Enough?

HTML is an instance of SGML. However, HTML doesn't have a DTD, or, more accurately, has only one DTD that is unchangeable. Since HTML could not be extended, Web powerhouses Netscape and Microsoft added tags to enhance their browsers. These tags were incompatible with competitive offerings and led to a situation of mass confusion as Web developers were forced (for a short while) to back a single browser when developing their Web pages.

Furthermore, HTML is often poorly implemented. For example, end tags are not always required, null tags are not clearly marked, and so on. This is as much the fault of the specification as of Web page developers. This means that Web browsers require a huge amount of code to guess what the HTML was supposed to represent.

The Value of XML

XML, on the other hand, is simpler to implement—the spec is less than 40 pages. Despite this simplicity, XML has fully separated format from content. Unlike HTML, XML is a full-featured SGML implementation, minus the difficult-to-use features that were not all that necessary for the ordinary business environment anyway. The phrase used in the industry is that XML is more like SGML-- than like HTML++.

Valid Versus Well-Formed XML

There are two types of XML documents: valid and well-formed. A valid XML document is exactly the same as any valid SGML document—all the tags are defined in the DTD, they appear in an appropriate order, there are corresponding end tags for all begin tags, and so on. This will be the normal type of XML document used for electronic information exchange.

A well-formed XML document includes properly constructed tags, but there is no DTD associated with it. This is similar to HTML and permits a simpler process in building an XML page when the content of the page is not critical.

Why Is XML Important To Me?

Web pages can have an amazing variety of things happening. Besides the underlying page itself there maybe three-dimensional spinning objects, marquees scrolling across the bottom, you name it! And what is the purpose of all this activity? To attract your eye.

But how many Web sites can you reasonably ever expect to see? Only a fraction—and each day you're falling farther and farther behind.

In the future, you'll find what you need on the Internet through the use of intelligent agents. Intelligent agents, or know-bots, are software engines which browse the Internet looking for the information. These products will grow to be far more sophisticated than just search engines. They will be able to read Web pages just as you do—in human language. XML is an integral part of making these intelligent agents effective.

Think about all the flash and trash on Web sites—the scrolling text, the audio, the spinning icons. To the intelligent agents, this is just background noise—simply an obstacle to getting to the real information. In the future, we'll print information for two audiences: human readers and machine readers. We'll be concerned with formatting for the human and with tagging for the computer.

Let's say that your job is to print bank statements. To present the bank statement on the Internet, you could translate your print data (AFP, Metacode, PCL, etc.) to HTML. For your human reader, this is adequate.

Shortly, however, your human reader will want PC software to read the bank statement and import and store data. The software which reads your statement could care less about the format, but wants to understand the meaning, that is, the author's content.

The following shows a simple bank statement. In it, we see check numbers, dates, and amounts.

Account Number:	1234567890	
Checks	Date	Amount
———	———	———
100	01/01/98	123.00
101	01/02/98	234.12
102	01/11/98	500.00
Current Balance		4,345.00

In a presentation datastream, we would see the data something like this:

X, Y, font, orientation, Account Number
X, Y, font, orientation, 1234567890

X, Y, font, orientation, Checks	Date	Amount
X, Y, font, orientation, ——	——	———
X, Y, font, orientation, 100	01/02/98	123.00
X, Y, font, orientation, 101	01/02/98	234.12
X, Y, font, orientation, 102	01/11/98	500.00
X, Y, font, orientation, Current Balance		4,345.00

Of course, this print data does not have to be in this order. For example, the fifth record (100 01/02/98 123.00) could actually be three different records, each with its own X and Y address. And since this is true, there's no reason to assume that the data value 100 will be found in the data before the date value but after the text string Checks. You can't really predict anything at all about the order of print data which makes it problematic for software to understand which data are which.

In the XML example above, we can clearly see what the purpose of string 100 is - it's a check number, because it follows the XML tag <check-number>. This is easy to determine no matter where the string 100 appears on the page or what it appears next to in the print data.

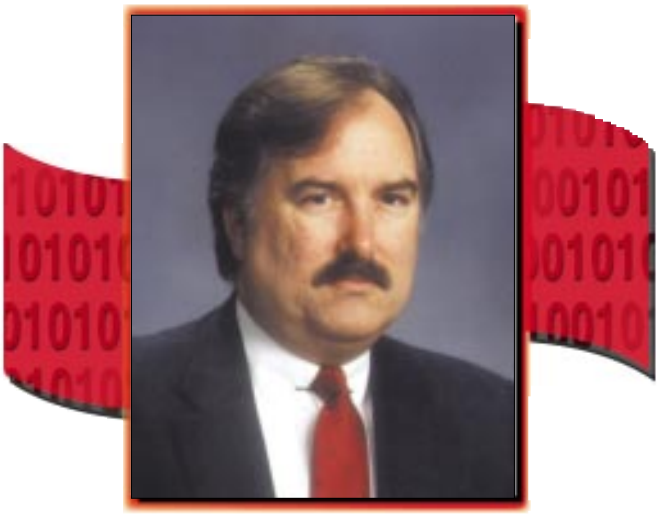
Bank Statement in XML:

<statement><january><text>Account Number:</text><account-number>1234567890</account-number><heading-text>Checks</heading-text><heading-text>Date</heading-text><heading-text>Amount</heading-text><underscore>———</underscore><underscore>———</underscore><underscore>———</underscore><check-number>100</check-number><check-date>01/01/98</check-date><check-amt>123.00</check-amt><check-number>101</check-number><check-date>01/02/98</check-date><check-amt>234.12</check-amt><check-number>102</check-number><check-date>01/11/98</check-date><check-amt>500.00</check-amt><total-text>Current Balance</total-text><total-bal>4,345.00</total-bal></january></statement>

XML: The Future Is Here

The sort of data exchange enabled by XML is already happening today. OFX (Open Financial Exchange)—the format used by Intuit Quicken and Microsoft Money to talk to banks, and CML (Chemical Markup Language)—which exchanges information on chemical formulas—are just two applications in use today.

We will continue to generate print streams for paper printing and archiving. In the short run, we will see software which enables the conversion of legacy print streams to XML. In the long run, we will create the XML directly and create legacy print streams from XML as needed for paper.



William J. "Bill" McCalpin, EDPP, is senior architect for The Xenos Group, a software vendor and systems integrator based in Toronto, Ontario. Bill has been active in Xplor since 1986 and is currently an associate editor of Xploration. He can be reached at billm@xenosgroup.com.

XML Resources

• www.w3c.org — the official World Wide Web Consortium site (you'll find links to the XML spec here). As you research this site, you will see the name of Tim Bray, one of the members of the committee who developed the XML specification.

• www.developer.netscape.com/viewsource/bray_xml.html. Here you'll find an excellent (and brief) discussion of XML by Bray. In addition, there are other references to SGML both at the w3c Web site as well as www.oasis-open.org/cover/sgml-xml.html.