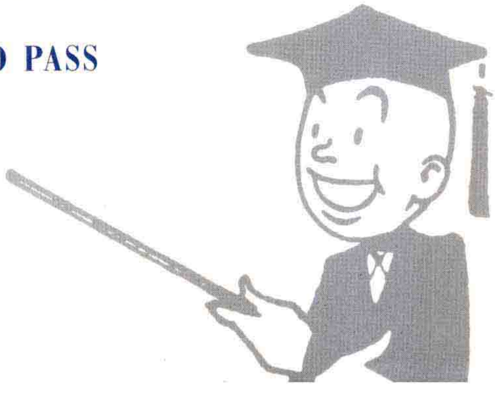
More than 30 million Schaum's Outlines sold!

XML

SCHAUM'S easy OUTlines

CRASH COURSE

- INCLUDES FULLY SOLVED PROBLEMS FOR EVERY TOPIC
- EXPERT TIPS FOR MASTERING XML
- ALL YOU NEED TO KNOW TO PASS THE COURSE



SCHAUM'S *Easy* OUTLINES

XML

BASED ON SCHAUM'S
Outline of Theory and Problems of XML
BY ED TITTEL

ABRIDGEMENT EDITOR
DALE A. BROWN, Ph.D.

SCHAUM'S OUTLINE SERIES

McGRAW-HILL

New York Chicago San Francisco Lisbon London Madrid
Mexico City Milan New Delhi San Juan
Seoul Singapore Sydney Toronto

ED TITTEL has been an instructor at Austin Community College since 1996. He teaches markup languages and networking topics. The author of over 100 computer books and the originator of the Exam Cram certification preparation series, he also teaches various Window topics at the NetWorld + Interop trade show.

DALE A. BROWN has been a Professor of Computer Science at The College of Wooster since 1987, teaching courses in introductory and intermediate programming, computer organization, database technology, machine intelligence, and theory of computation. He received his masters and Ph.D. degrees from Syracuse University.

1 2 3 4 5 6 7 8 9 0 DOC DOC 0 9 8 7 6 5 4

ISBN 0-07-142245-5

SCHAUM'S *Easy* OUTLINES

XML

Contents

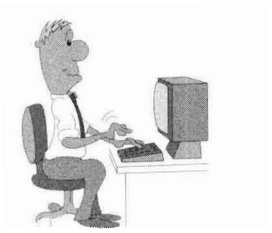
Chapter 1
AN OVERVIEW OF XML

IN THIS CHAPTER:

- ✔ *Origins of XML*
- ✔ *Differences between XML and HTML*
- ✔ *Uses of XML*
- ✔ *XML Document Structure*
- ✔ *DTDs*
- ✔ *Schemas*
- ✔ *Validation*
- ✔ *Namespaces*
- ✔ *Comments*
- ✔ *Processing Instructions*
- ✔ `xml:lang` *and* `xml:space`
- ✔ *XML tools*

Origins of XML

The Extensible Markup Language (XML) emerged in 1996 as a subset of the Standard Generalized Markup Language (SGML). The design goals in the original World Wide Web Consortium (W3C) XML Working Draft document described a software-readable markup language that integrated easily with existing markup languages, such as HTML and SGML, and was easily readable by humans. In February 1998, XML 1.0 reached W3C Recommendation Status. A January 2000 update of HTML incorporated some basic features of XML with the release of the Extensible Hypertext Markup Language (XHTML).



What makes XML so special? The key lies in XML's name itself—the Extensible Markup Language. The word *extensible* reflects that the language is flexible, scalable, and adaptable. One can "extend" the information in a set of data by defining new tags that identify particular components in that data. XML data can take whatever form is necessary for distribution of information, either over the Web or between software applications. The constraints and limitations of HTML do not apply.

Remember

The term *markup* describes a process of identifying the different elements of a document so that a program or person can process them. For instance, web designers use HTML markup tags, like `<br>`, `<head>`, and `</head>`, to identify elements for interpretation by a browser.

XML provides two means of establishing new tags to identify elements of the data: Document Type Definitions (DTDs) and schema documents. By coupling XML documents with a DTD or a schema, it is possible to define markup tags (or element identifiers) in unique and specific ways. One can customize such tags to meet a particular need and structure while enabling information exchange in a language through which both software and humans can easily interact. By creating rules and elements to meet the needs of the task, XML allows the exchange of structured data in an accessible yet easy-to-read format.

Data searching and retrieval is an area of tremendous growth and research regarding HTML. As anyone who has used search engines knows, search phrasing is somewhat a "dark science." It seems that searches frequently recover either too little or too much information, depending on the user's skill, luck, or patience. The fundamental problem is that a given word has many different meanings, depending upon the context in which it appears. Users need a way to identify the meaning of a word by knowing both its spelling and the context in which it appears. In HTML, one uses meta elements to mark keywords, dates, and so on, but this method is weak because of the small number of available meta elements and the unlimited variations of data that can occur. By contrast, XML permits the definition of new tags to match any new type of data. It therefore enables context-dependent searching.

The designers of XML had several initial objectives. They desired:

- A content-driven language derived from and compatible with SGML
- Tools for Electronic Data Interchange (EDI) and other data-driven applications that HTML lacked
- Platform-independence
- The capability of distributing data over the Web through the browser
- The capability of distributing data over the Web through means other than the browser.

Differences between XML and HTML

Although HTML, XML, and (their hybrid) XHTML all use markup tags as containers for data elements and appear on the surface to be quite similar, the tags themselves are quite different, not only in definition and meaning, but also in their methods of creation and specification. Whereas HTML and its successor, XHTML, use elements that are more or less universally defined and accepted as HTML 4.01 or XHTML 1.0 (via the implied DTD that the browser includes), XML allows, encourages, and thrives on elements that some user or group has created for structuring data to a specific intent and purpose. If an element is required for XML and is not part of the DTD, the author can create the element needed and define it as an add-on.

Note!

Don't confuse *tags* and *elements*. The information:
`<title> Outline of XML </title>`
is an element while `<title>` is the opening tag and `</title>` is the closing tag.

At the core, the differences between HTML and XML are very simple: HTML is a *presentation* markup language, readable and rendered by almost any modern Web browser, whereas XML is a *content* markup language, with no inherent, or built-in, presentation elements, only content-definition elements.

XML allows one to design elements as an application-specific process, defined in a schema or DTD, and used over the Web in a language that best describes the data itself. Definition of the necessary XML elements accomplishes a number of otherwise difficult tasks of context clarification. Again, XML defines the data, and HTML defines the presentation. To put it another way, XML processes data, whereas HTML displays data.

As even a casual user of HTML knows, most of the HTML elements influence the layout and look of the document—the presentation. Web professionals are all very familiar with such elements as `center` or `font`. These elements (and other presentation elements) do not indicate the type of data contained. Only a few HTML elements do this. For Web pages, this is generally adequate, and use of these elements will continue to grow and thrive as the Web continues to expand. However, for indicating the type of data or their purpose, HTML falls a bit short with its minuscule number of content-related elements.

You Need to Know ✓

HTML combines information about both *content* and *presentation* format. XML consists only of content information. If it is necessary to present the data, presentation format is the responsibility of other support software.

Because XML is a content markup language, it is necessary to specify only the actual data contained in the tags. For example, one might wish to create a list of books that consists of the title, the author, the subject, the publisher, and more. There are no HTML elements to specify that the enclosed text is the name of the author.

XML does not have these limitations. The document's creator is relatively free to create the needed elements in a DTD or a schema that the reading software or agent can access. The story does not end here, however. Because XML is content-driven, it is necessary to provide a separate method for presentation. Later chapters discuss this fully. Suffice it to say here that one combines the XML document with a *style sheet* via the Extensible Stylesheet Language (XSL) or Cascading Style Sheets (CSS). The style sheet defines how to present the content of each element. The best part is that, for software or agents that do not require presentation markup, one can simply provide the XML and the content it contains. Simple exchange of XML data does not necessitate presentation.

Note!

XML requires that the author create *well-formed* documents; that is, documents that properly match starting and closing tags and abide by logical rules of nesting. Most XML software will also *validate a* document by checking to make sure that the document uses tags only in a manner that is consistent with its DTD or schema.

Here are some of the key differences between XML and HTML:

- XML is a *content* markup language; HTML is a *presentation* markup language.
- XML allows user-defined elements; HTML elements are predefined.
- XML usually requires validation; in HTML, almost anything goes.
- XML is data-driven; HTML is display-driven.
- XML allows data exchange between software applications without the need for presentation information; HTML is designed for visual presentation.
- XML is strictly defined and interpreted; HTML is very loosely interpreted. XML elements must be closed; in HTML, empty elements do not need to be closed.
- XML is case sensitive; HTML is not.

Uses of XML

Early uses of XML include documents for traditional Web browsers and industry specific document exchange. Industries and disciplines such as chemistry, mathematics, real estate, weather observation, banking, electronic data interchange (EDI), and many more offer a vast potential use of XML. Because a DTD or schema defines the XML document to be specific to the needs of the users, XML can easily support any type of industry. For example, in the weather observation industry, the DTD or schema can provide the elements needed for wind speed, pressure, temperature, humidity, and so on. Observer systems can record the data, format it according to the element definitions, and forward it for interpretation by forecasting systems. Because each element defines a particular type of content and its formatting rules, the documents are mutually understood and easily readable by a human. If the XML ultimately requires presentation by a Web browser or other agent, the person responsible for presentation specifies the necessary XSL or CSS style sheet as well as rules for presentation; this encourages strictly defined elements and very narrow application within a specific industry or academic field. Many DTDs and schemas already exist for these applications, and others are being created constantly. XML will continue to evolve as more and more uses are discovered and implemented.

Wireless applications press XML into service in novel and very practical ways. The need to conserve bandwidth for and format information on a small, handheld screen cries out for simple, customizable data exchange media. XML can pass pure data or provide minimal presentation markup without all the excess baggage of HTML. Many flavors of XML will develop to meet the very narrow criteria of wireless display but expand into fully presented Web pages when the application requires it. The XML concept of separation between content and presentation is ideal in such circumstances.

XML Document Structure

Previous sections have emphasized that XML describes content rather than format. An XML document holds information, not a description of how to display that information. However, the information takes two different forms:

- *Logical structure* defines the units and subunits of the data containers (the elements) and the components they contain.
- *Physical structure* provides the data that goes into the elements. They include text, images, or other media, as allowed by the logical structure.

Consider the following subset of an XML document:

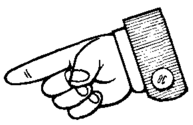
```
<letter>
      <greeting>
      Hello there
      </greeting>
</letter>
```

It forms a letter that contains a greeting that, in this case, consists of the string "Hello There." The tag information (`<letter>`, `<greeting>`, `</greeting>`, and `</letter>`) describes logical structure, while the string, `"Hello there,"` represents physical structure. The physical structure is the raw data, and the logical structure adds meaning to the physical structure by establishing its context. Without the logical structure in the tags, the characters are simply a string. With markup, it constitutes a greeting that is part of a letter.

The addition of more logical structure to the above markup generates a complete XML document:

```
<?xml version="1.0" standalone="yes"
      encoding ="UTF-8"?>
<letter>
      <greeting>
            Hello there
      </greeting>
</letter>
```

The new markup indicates that the document is using XML 1.0, that there is not an external DTD (`standalone="yes"`), and that the document is using `UTF-8` encoding for its characters. Following this *XML processing instruction* is the *opening root tag* (`<letter>`), which is followed by the rest of the document and the *closing root tag* (`</letter>`). Components, `version` and `standalone`, are *attributes* of the `xml` markup. Attributes specify property values of the elements within whose markup they exist. They have both names and values. In this case, the `xml` element has value `"1.0"` for the attribute with name `version` and value `"yes"` for the attribute with name `standalone`. As in the example, an XML



document specifies an attribute's value in the opening tag for the element that contains it. For a document to be well-formed, attribute values must have quotation marks around them.

The physical structure of an XML document is the structure of the actual data and information. XML documents that share a logical structure, as defined by the schema or DTD, can vary dramatically in physical structure, based on the data they contain. In the preceding example, the logical structure `<greeting...</greeting>` could contain "Hello there" or some other greeting as the physical structure.

Notice the nesting of the elements in the document. The `<greeting ... greeting>` tag pair nests within `<letter>...</letter>`. This careful attention to the nesting of elements is part of what makes a well-formed XML document.

Consider the addition of a few simple elements to the example:

```
<?xml version= "1. 0" standalone= "yes"
   encoding= "UTF-8" ?>
<letter>
   <greeting Hello World! </greeting>
   <signature>
      <message>
      Most Sincerely...
      </message>
      <name>
      Joe Smith
      </name>
   </signature>
</letter>
```

Again, notice the nesting as emphasized by the indentation.

The `letter` contains both a `greeting` and a `signature`, and `signature` contains both a `message` and a `name`. Therefore, this is a well-formed document. If any of the element tags are out of sequence, for example, if the closing `message` tag and the opening name tags are transposed or overlap, the document is not well-formed. As a further requirement of being well-formed, all attribute names should be unique. Finally, all elements should be closed, whether in pairs, as in this example, or as empty tags, such as `<subject name= "example XML"/>`. Notice the slash and closing bracket (/>). This creates a single but complete tag. The familiar `<br>` tag in HTML would now take the form `<br / >` to be correct in XML. Note that for backwards compatibility reasons, there must be a space before the closing slash in XHTML. An example would be `<br / >`.

Note!

XML and its related software applications ignore indentation. The illustrations in this outline use indentation simply to enhance the reader's understanding of the markup.

Another important aspect of XML document structure is the use of *entities*. These represent reserved sequences of characters used to distinguish regular text from markup. For example, the left angle bracket (<) identifies the beginning of a tag in markup. Therefore, if the goal is for the parser (browser) to display a left angle bracket and not render it as markup, it is necessary to use the bracket entity. In this case, the correct entity is `&lt;`.

Entity references begin with an ampersand (&) and end with a semicolon (;). To use an entity, one can reference it by its unique name. Entity declarations enable the association of a name with some other fragment of content that can be part of regular text, part of the DTD, or a reference to an external file containing either text or binary data. The current discussion deals only with predefined entities. Table 1-1 lists the five predefined entity characters.

Table 1-1 The Five Predefined Entity Characters

HTML	Character	Description
`&gt;`	>	Greater than
`&lt;`	<	Less than
`&amp;`	&	Ampersand
`&apos,`	'	Apostrophe
`&quot;`	"	Double quote

These entities serve in instances where their corresponding symbols represent themselves rather than markup symbols. Of course, it is possible to declare other entities, but these are the only ones built into XML.

You Need to Know ✓

The basic rules of creating well-formed documents are:

a. Begin the XML document with a declaration.
b. Provide at least one element (the root element) that contains all other elements.
c. Nest tags correctly.
d. Use both start and end tags for elements that are not empty, and close empty tags properly.
e. Quote all attribute values.

Unless a document is well-formed, it will not parse correctly, and it can cause an error on the parser; therefore, well-formed XML should be a fundamental goal at all times.

DTDs

A Document Type Definition (DTD) can provide element meaning, attributes, logical structure, and context. It extends logical structure definition beyond the minimal standards of well-formed documents. A DTD defines the elements and their attributes that specialize XML to specific disciplines and uses. It also enables exchange and accessibility by other software. A DTD is a set of rules that explicitly define the name, content, and context of each element. Therefore, a DTD can serve as the foundation of an XML document, defining the template for processing images, links, and other entities.

The DTD defines the building blocks (elements) that the XML document can use. It is possible to declare a DTD in-line as part of the XML