Chapter 1

Understanding What XML Is — and Why You Should Care

In This Chapter

- ▶ Examining and explaining XML lingo
- ▶ Attending a short history course
- ▶ Introducing the many uses of XML
- ▶ Deciding what to use to build XML documents

Have you ever wanted a document format that you could use to exchange data across the Internet? Well, eXtensible Markup Language (XML) may be just the solution for your problems. Various software products in many different industries widely employ XML to present and organize their data.

XML is a programming language based on the Standard Generalized Markup Language (SGML). The flexibility of XML has made it a necessity for exchanging data in a multitude of forms. Accessing information with XML is so much easier than with HyperText Markup Language (HTML). For example, with XML, you can send the same information to both a person using a mobile phone and a person using a Web browser. In addition, you can customize the information to be displayed appropriately on the various devices. Welcome to a new age!

Getting started with XML isn't difficult. Just check out this chapter, and you'll find out what markup languages are, what XML is, what you can use XML to do, and what types of XML applications exist — waiting for you to take advantage of them.

Getting to Know Markup Language Lingo

Although having a working knowledge of HTML is a good idea when getting started with XML, you don't have to be a markup pro to read this book or to use XML. If you're new to the markup world or if you need to brush up on your vocabulary, the following list should help you out.

These terms are the most common ones that you need to become familiar with to understand markup languages in general (including XML):

- **Attribute:** In XML, a property associated with an XML element that's also a named characteristic of the element. An attribute also provides additional data about an element, independent of element content.
- **Document Type Definition (DTD):** An SGML-based statement of rules for an XML document that specifies which elements (markup tags) and attributes (values associated with specific elements) are allowed in your documents. A DTD also governs the order in which the elements and attributes may or must appear.
- **Element:** A section of a document defined by start and end tags or an empty tag, including any associated content.
- **Metalanguage:** A language used to communicate information about a language itself; many experts consider both SGML and XML to be metalanguages because they can be used to define markup languages.
- **Nesting:** An ordering of elements whereby a child element is opened and closed before its parent element is closed (child elements nest within parent elements).
- **Schema:** An XML-based statement of rules that represents an XML document's data model and defines its elements (or objects), their attributes (or properties), and the relationships between different elements.
- **Syntax:** The rules that govern the construction of intelligible markup fragments.
- **Tag; empty tag:** The markup used to enclose an element's content. An empty element employs a single tag; a regular element (which isn't empty) has an opening and a closing tag.
- **Valid:** A markup language document that adheres to the rules outlined in an associated DTD or schema document.
- **Well-formed:** A markup language document that adheres to the syntax rules for XML, which are explicitly designed to make that document easy for a computer to interpret.

A Wee Bit of Background: Markup Languages

A *markup language* is a language used to label, categorize, and organize data or document content. *Markup* is what describes document or data structure and organization. *Content,* such as text, images, and data, is what markup contains and is also generally what's of greatest interest to humans who read or interact with data or documents.

XML is a markup language and so is HTML. In fact, both of these markup languages derive from the same parent: SGML. And eXtensible HTML (XHTML) is a reformulation of HTML using XML. Figure 1-1 shows you a diagram of the happy markup family tree.

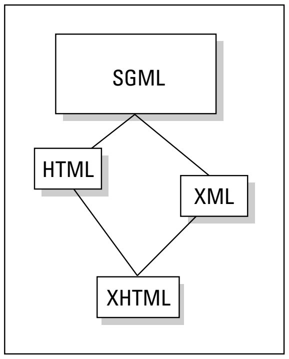

Figure 1-1: Markup languages are related like this.

Introducing SGML: The mother of all metalanguages

SGML is a powerful metalanguage; its primary job is to define other markup languages, including HTML, XHTML, and XML itself. In other words, SGML is the parent of HTML, XHTML, and XML, among many other markup languages. (Refer to Figure 1-1.)

SGML is a "kitchen sink" metalanguage: That is, it incorporates everything but the kitchen sink — all the facilities and capabilities that are intended to make it as complete as possible. These extra bells and whistles, however, make SGML software difficult to implement and slow and unwieldy to run.

The purpose of SGML was to create a powerful, general-purpose tool for defining markup languages that could work faster and better on more focused tasks. XML is a markup language that defines a carefully chosen subset of SGML meant to work efficiently to better support the World Wide Web (WWW) and other networked applications across the Internet.

Introducing HTML

HTML is a markup language that was originally developed to describe and deliver textual documents across the Internet. HTML tags define only basic text elements, such as paragraphs and divisions, and offer only limited presentation controls for text.

HTML is not a general-purpose tool to label and organize text (or data). It uses a predetermined set of markup elements that can be expanded only by agreement and alteration of HTML's underlying markup language description. This makes HTML a *closed* markup language in the sense that adding new markup elements is not part of its built-in capabilities.

The closed nature of HTML explains why it has gone through so many versions. Starting with the public introduction in 1993 of HTML 1.0, up to its "final definition" as HTML 4.01 in 1999, each of its new incarnations represents incorporation of new markup and capabilities to meet document designers' ever-expanding desire for more functionality.

Introducing XML

In a very real sense, XML is the child of dire necessity because the custodians of HTML realized that a closed markup language couldn't be maintained and still meet demands for document- and data-handling on the Web. With daily demands for change and schedules that led to an average interval of about 1.5 years between HTML versions, its keepers simply couldn't keep up.

XML was developed by a working group under the umbrella of the World Wide Web Consortium (W3C). To solve the problem at its root and to create an *open-ended* markup language that could easily accommodate future expansions and additions, the W3C created a working group to define such a markup language. Officially called the XML Working Group, this gang consisted of researchers and experts from many parts of the computer industry, including Internet and intranet technology gurus, publishing wizards, and markup language mavens. The primary goal of this merry band of technologists was to bring the kinds of capabilities found in SGML to the Web, without all the extra bells, whistles, fat, and calories. Rather than calling it *SGML Lite*, they called it *XML*.

By February 1998, version 1.0 of the XML specification was unleashed on the world, and the XML markup revolution began in earnest.

Introducing XHTML

XHTML closes the loop between XML and HTML. The W3C calls XHTML "a reformulation of HTML 4 in XML 1.0." In simpler terms, XHTML represents the latest incarnation of HTML, using XML markup and syntax to re-express HTML in XML terms. Because XML is open-ended, by recasting HTML in its terms to create XHTML, XHTML is open-ended, too. That's why the *X* in XHTML stands for *eXtensible*.

In its simplest application, XHTML uses markup that differs only slightly from its HTML predecessor and works as well as HTML in the vast majority of Web browsers. More ambitious applications of XHTML take advantage of the *X*, so to speak, adding markup definitions to the basic set originally defined for HTML 4.0 so that document developers can expand the set of markup elements that they employ within their XHTML documents.

Understanding Why XML Suits So Many Dang Applications

If you take a close look at today's Internet industry and at the many different research and development initiatives underway, you soon recognize that pinning down a single, definitive function for XML is nearly impossible. The W3C created XML as a way to disseminate complex, structured data and documents over the Web. In other words, the open-ended nature of XML is what makes it so useful for many different things and so difficult to put into a single, small box.

In fact, case studies of XML never fail to mention new and exciting possibilities where XML adds value to existing environments or solves previously intractable problems. That's probably why XML applications have been defined and are widely used for everything from chemical formulas to genealogies (family trees). Nonetheless, many key XML implementations fall into one of two categories:

- **Complex document creation,** such as a large, multisection description of a software application, with an auto-generated index, table of contents, and working code examples.
- **Data exchange,** where complex data structures, such as patient medical records, can be completely and comprehensively described and where such data can easily be reformulated to view in a report, print in a document, or imported into a database.

Experts in many applications and industries have already created specific markup languages that you can use as is. We describe a few popular examples in Chapter 22.

The universe of XML-based markup languages, also called *XML applications*, is constantly expanding. The odds are high that you can find a markup language suitable for your particular needs. If the language doesn't completely satisfy you, extend it — after all, XML is the *eXtensible* Markup Language.

Using XML for a Variety of Output Options

Authors of XML-based Web documents don't make any assumptions about how the documents will be used on the client side. HTML pages, on the other hand, are designed for one particular purpose: to display information inside a browser. Browsers process HTML documents easily, but software sometimes has a difficult time post processing the information that such documents contain. This limitation doesn't apply to XML documents.

The phrase *post processing* means taking information delivered within a document and using that information in some other process or program. For example, suppose that you receive a purchase order in the form of an XML document. An application that understands XML purchase orders can use that data to determine which items in what quantities have been ordered and can even send instructions to another piece of software to generate a pick-list so the order can be picked, packed, and shipped from the warehouse. Now that's what we call post processing! To achieve the same goal in an HTML-based environment, you'd have to write a special purpose program to read and interpret the HTML document. Comparatively, with XML you can use standard tools to parse and process document contents directly — with no extra markup needed!

XML is not a replacement for HTML. XML enables the Internet industry to invent a new set of powerful tools for many purposes.

In many cases, XML documents are used with style sheets to provide high-quality output on-screen. You can use the same data, however, to send information to a speech-synthesis program that reads the text to a person who is vision impaired. Alternatively, that same data might also create output on a Braille reader. The same document with a layout program and a style sheet also might be used for high-quality printouts. See Figure 1-2 for an illustration of this post processing.

The beauty of this concept is that you never need to change the XML data to create output for different devices. You need only to use different pieces of software that can provide the output for a particular format or output device.

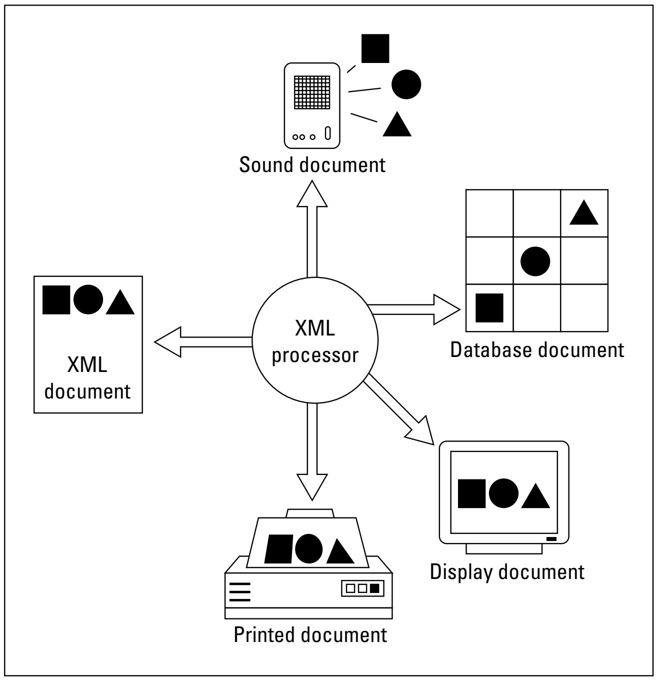

Figure 1-2: Use XML for different outputs.

Using XML for data and HTML for display

You may often see a combination of XML and HTML in everyday online use. XML is all about preserving information. HTML is all about, or at least is accomplished at, displaying information within a browser. Why not combine the best of these two worlds? The basic idea is to have the original data based on XML, where all the rich markup and additional information about a document is available. Then you can use the same document for many different purposes and use the "intelligence" in the data to build powerful applications to display the data on-screen and translate it into HTML, a concept illustrated in Figure 1-3.

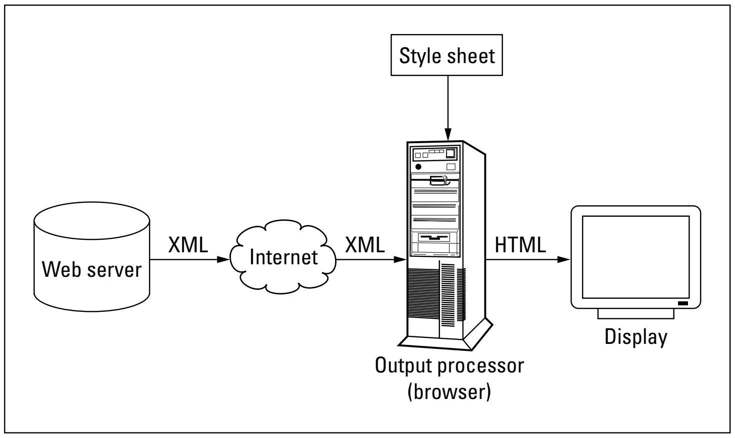

Figure 1-3: Translate XML to HTML on the client side.

XML stores and organizes the data, and HTML renders it inside your browser by using a style sheet. The methods differ depending on when the XML data is translated into HTML for display inside a browser, as well as on the type and version of the browser.

The conversion experience can occur in the following ways:

- XML data is sent to the client (your browser), and your browser uses a style sheet — extra information that helps your browser make XML look interesting — to display HTML.
- The XML data resides on the server where it passes through an XSL Transformation and is converted to HTML before being sent to a browser.

Using XML for enhanced post-processing functions

HTML is a display-oriented language. That's its primary function, plain and simple. And that's why HTML isn't the best option for post-processing functions, such as constructing searchable local indexes. You can see the poor-quality results of HTML's lack of post-processing capability when you query a search engine to find Web pages on a particular subject. Very often, you get back either 0 hits or 100,000.

The XML Files and other XML phenomena

Those in the W3C's XML Working Group keep most of their materials, including text for speeches, presentations, and technical specifications, freely available on the W3C Web server. Visit this site to find out what you need when you need it most — that is, when you're slaving over some hot XML markup. The address is

```
www.w3.org/XML
```

Not only have these XML gods declared their collective wills, but they've also started a regular e-zine called — you guessed it! — *The XML Files.* Keep up with the latest and greatest information on XML and its development by visiting

```
www.gca.org/whats_xml/whats_xml
    _xmlfiles.htm
```

In addition, you may want to visit `www.xml.com` and click the Annotated XML 1.0 link to take a peek at Tim Bray's annotated version of the specification.

Because XML data preserves both context and semantics, building applications that apply some smarts to electronic documents is a lot easier than it would be in HTML. Suppose that some XML documents contain an `Author` element. Because you can search any XML document to elicit the content of specific elements, this means you can search the Web for all XML documents in which *Ed Tittel* appears in the `Author` element. XML makes it possible to focus on any specific piece of information within a document because it can identify and access individual document elements.

Likewise, XML enables search engines to access information inside documents because those engines can use XML element contents and attributes to find specific entries or text. So instead of relying on the document titles and headings that robots usually gather from HTML documents to index their contents, they can access all the content (and markup) inside XML documents and get more accurate results. Imagine actually being able to use the results of a search on the first try!

Using XML for data exchange

Suppose that you want to exchange database information across the Web. In particular, suppose that you want to use a Web browser to send information from a user questionnaire back to a Web server. To accomplish this and many other tasks, you need a document format that's extensible, open, and nonproprietary. An *extensible* format is one that can be tailored or customized for specific applications. An *open* format is one that's well documented and widely available to would-be users. And finally, a *nonproprietary* document is one that's expressed in an accepted or standard form of notation that isn't the exclusive property of some individual, company, or organization. These characteristics are what make it possible to adapt to changing conditions, to leverage the work of others, and to avoid extra expense or legal liability.

You could use import/export filters to interchange file formats, but XML is the best solution for data exchange because XML can constantly be improved upon to provide a single platform for data interchange between multiple applications.

Building XML Documents

Regular old-fashioned editors such as Notepad will do the job if you're just getting your feet wet with XML, but text editors that are built specifically for XML are the way to go if you plan on using XML regularly. These editors often look like a blend of traditional word processors and HTML editors. In fact, most XML editors work so much like word processors that you could easily forget that you're working with XML.

XML editors can make your job easier and help keep those creative juices flowing! (Tracking tags and cleaning up structures can interrupt, and sometimes completely destroy, the creative train of thought.) XML editors have two distinct features that are essential for creating good XML documents:

- **Ease of markup:** XML editors, such as XML Spy and XML Pro, can add markup to text as simply as you can turn text **bold** in today's word processors. All XML editors provide the capability to select text with a cursor and choose which markup you want to apply from a menu of selections. (See Chapter 20 for more on XML Spy, XML Pro, and other XML authoring tools.)
- **Enforcing document rules:** For many applications, XML editors can determine which element types can appear in certain contexts. In this way, the editor helps you avoid making syntax or structure mistakes. For example, if you specify that the `ChapterTitle` element is valid only at the beginning of a chapter and never within an ordinary paragraph, the editor can make sure that your rule is enforced if you accidentally break it.

XML is a subset of SGML, so many authoring tools and editors previously used for SGML have been recast and are now ready to take on XML.

Breaking Out Your Text Editor

To create your own XML documents, you don't need anything more than a plain old ASCII-based text editor such as Notepad. If you plan to use XML more often than once in a while or for a lot of pages, however, think about getting an XML editor.

If you're using Windows, you can access Notepad by choosing Start➪Programs➪Accessories➪Notepad. A new Notepad window opens. You can save the files just as you would in a word processor and do simple functions such as copy and paste. Aside from that, though, Notepad is a pretty bare-bones program.

Check out Chapter 2 for steps on creating an XML document by hand. But before you skip on, here are some handy text-editing tips:

- **Be specific with filenames.** When you save a document for the first time, most text editors have predefined settings for the document's extension. If you use generic XML, just use `.xml` as the extension. With a specific XML application like Channel Definition Format (CDF), for example, use `.cdf`.
- **Beware of binary.** You can create XML documents by using one of the more sophisticated document-authoring programs such as Microsoft Word or Corel WordPerfect. Keep in mind, however, that the native File➪Save format of such programs is a binary format. For example, the native and default file format for a Word file is the `.doc` format. That format is not suitable for XML (or HTML). Make sure to save your XML document as a text file (`.txt`) if you use one of these programs.
- **Hand coding means you have to do a lot of typing.** You must insert all markup yourself when using you use a text editor such as Notepad. If you have a paragraph such as

  ```
  This section is about XML editing
  ```

  you need to type the required markup tags by hand. If your XML markup for a paragraph was `Para`, your text would look like this:

  ```
  <Para>This section is about XML editing.</Para>
  ```