

XML for beginners

Reviewing **Learning XML: Creating Self-Describing Data**, by Erik T. Ray, O'Reilly

Introduction

I had seen the term **XML** many times, but didn't have a clue what it was other than some glorified HTML, so when Charlene asked for someone to review this book, I jumped at the chance. O'Reilly, the publisher, had given her a couple of free copies to kick off the new IESTC member-benefit: a 20% discount off any purchase of O'Reilly books from their website. Just sign on to the My IESTC section of <http://www.iestc.org> in order to access the O'Reilly website with the discount.

I was pleased that **Learning XML** is well written and easy to follow, even for a non-programmer like me. However, I can't say I've actually *learned to program* XML by reading the book. I would have thought that **Learn about XML** would have been a more appropriate title. If you really want to learn how to program XML, check out O'Reilly's ample selection of XML books for further study.

The author, Erik Ray, is working with O'Reilly to develop ways to use XML in their publishing business. He even used XML as the authoring tool to write this book.

The main problem with **Learning XML** is that it was published a year and a half ago, which means that some of the material has probably been updated since then. Erik Ray being aware of this, included a large reference section in the appendix to help you keep up-to-date. The first he mentions is a good, comprehensive website at <http://www.xml.com>.

In order to avoid a mess of different standards, the wise people at the World Wide Web Consortium (<http://www.w3.org>, which tries to keep track of developments in web programming) are also monitoring XML, so we don't end up with mutually incomprehensible dialects. They have recently approved the newest standard for XML 2.0, although Ray's examples, being 1 ½ years old, use XML 1.0.

OK, so what is XML?

The closest I can come to explaining XML (Extensible Mark-up Language) is that it's sort of like combining the codes you use for mail merging with HTML. Instead of marking up a document for formatting, as you do in HTML, you mark it up to show functions and categories.

An XML document is organized in nested *elements*, marked with *type* names, which can have a number of *attributes*. You can create any type or attribute you want, even "veggies". Later you can add a CSS (Cascading Style Sheet) for general formatting. That is where you can specify that any elements called "veggies" be formatted with a green font.

The mark-up can be used for much more than formatting, however. Read on!

So what good is that?

It's no big deal, of course, to format all your veggies green, even in WORD, by creating a green *veggie* character font. Anyone who has learned to use paragraph styles and style sheets in WORD or FrameMaker knows the advantage of marking up a text by sections.

What really makes XML flexible is that you can use the mark-up to *repurpose* a *single-source document* to display it as a print document or website, whatever you need. For this you use *templates* created in XSLT (Extensible Style Language for Transformation). It's up to you, which elements appear in the different document types, and you can provide special text or page design elements (like menu bars, a cover, TOC, index, warning symbols, or whatever) with the template. While you write the single document, you think about how it will be used in the different functions, and provide enough elements and attributes to describe it as you go.

Do I have to invent it all myself?

You can create most any kind of XML document if you want, but there are a number of *DTDs* (Document Type Definitions) available to guide you on your way. There's no sense reinventing the wheel when lots of people are out there fussing with it already. When you use an existing DTD, you can always define more elements and attributes to describe your particular data. DTDs operate with a concept of the *well-formed document*, so when you compile a document you are informed whether you've forgotten a tag or other syntax item. You can think of HTML as a sort of DTD used to define web-sites. But HTML is not used with a compiler, so you can't know if your document is well-defined before you try it out in a browser. Ray used a DTD called *DocBook* to write **Learning XML**. It was designed to provide the structure for working with technical documents. (See <http://www.oasis-open.org/docbook>.)

Another useful tool is *Namespaces*, which could be considered specialized dictionaries with predetermined categories. There is a math namespace, *MathML*, a chemical one, *CML*, and one for psychology, among others. Unfortunately Namespaces don't work well with DTDs which don't appreciate strangers messing up their well-formed documents.

If you've ever wondered what *Adobe FrameMaker with SGML* was, you'll rush out to buy it once you get into XML. *SGML* (Standard Generalized Markup Language – ISO-8879) is the granddaddy of all mark-up languages, including XML and HTML, so extended FrameMaker is **the** tool for us tech writers who want to get into the fun.

Of course, like HTML, you can write it all in a text editor, but I prefer being able to doctor a text that a high-end program has coded for me! There are also already a number of other programs you can use without all of the familiar FrameMaker features.

What's it look like?

The publishers have kindly provided a web backup to **Learning XML** at: <http://www.oreilly.com/catalog/learnxml> with a sample chapter and all the code examples, which you can download in a zip-file. Here is the first example of XML presented in the book.

```
<?xml version="1.0" ?>
<time-o-gram pri="important">
  <to>Sarah</to>
  <subject>Reminder</subject>
  <message>
    Don't forget to recharge K-9 <emphasis>twice a day</emphasis>. Also,
    I think we should have his bearings checked out. See you soon (or
    late). I have a date with some <villain>Daleks</villain>
    ...
  </message>
  <from>The Doctor</from>
</time-o-gram>
```

As you can see, the code sample starts by *declaring* the xml version in a set of tags with question marks. Then the document *type*, *time-o-gram*, is declared, with an *attribute*, *pri*. Different *elements* (*to*, *subject*, *message*, *from*) are marked with html-like tag sets, and there is an *in-line* element, *emphasis*, to mark separate words.

If you want to use specific vocabulary from a *namespace*, you just prefix the element with a reference to the namespace, which you declare with a link to the URL where it is located.

The most exciting aspect of XML, however, is the possibility of creating a nested **tree-structure** for the document. DocBook, for example, gives you a structure like this:

```
<book>
  <chapter type=preface>
    <title>
    <text>
    <section1>
      <title>
      <text>
    <section2>
      <title>
      <text>
    ...
```

When it comes time to format the document with a template or a CSS, you can refer to the different *nodes* in the tree, or even refer to the **siblings, parents** or **children** of a node, using expressions like `section2.title`. This enables you to format all the children of a chapter that have the attribute `title` using a particular font, or to skip all `section3` elements in a template for a Palm Pilot.

Learning XML provides the reader with numerous samples of XML, DTDs, templates, etc. with the necessary commentary. Particularly useful are two applications that are reused several times to demonstrate new material, so that you can see recognizable material in a new light.

Learning more

My 1500 words are scarcely enough to give you more than an impression of what XML is all about, so you will have to read elsewhere if you want to learn more. For this, I highly recommend **Learning XML** as readable, interesting and at a level that most of us can manage. It has definitely whetted my appetite for XML. I will be much more aware of XML applications when I encounter them and appreciate the flexibility XML provides the developer.

I have also noticed that the **STC** magazine **intercom** is running a column called **The Markup Language Guy**, by Paul H. Tyson, who will be educating us each month on the delights of XML.

A couple of small gripes

As tech writers we are probably more critical of each other's work than the general public is. Although I thoroughly enjoyed **Learning XML**, there were a couple of things that made reading it more difficult, particularly in the first chapters, while I was still fumbling with the concept.

I would have appreciated one more heading level to help me organize the material in my head. There were several sections in the beginning that presented far too many new ideas for me to keep track of without headings.

Furthermore, there were several examples that were not located on the pages where they were described, making it difficult to understand why they were there and how they fit into the greater picture.

Bonnie Yelverton, Upland, CA, August 2, 2002