An Introduction to XML

Richard Hill 2003

www.shu.ac.uk/schools/cms/teaching/rh1

Table of Contents

If you're viewing this document online, you can click any of the topics below to link directly to that section.

Section 1. Introduction

Getting Started

This tutorial gives you all the basics of the eXtensible Markup Language.

Read through each section in turn. If you need any software, everything will be explained.

From time to time you will come across some **exercises** to assist your learning. These activities will be presented in *italics*. You may be able to answer the question immediately, or you may have to find an answer from another source.

For now, all you need is Internet Explorer v6, and a text editor such as MS Notepad.

If you would like a Portable Document Format (PDF) version of this tutorial, click on one of the links at the top right of this window.

Don't forget to make notes as you progress through the tutorial. Some learners find it helpful to 'jot' notes into a text editor, running in a separate window, others prefer to write them down on paper.

To get started, click on 'Next'. To return to the beginning, click on 'Main Menu'.

Let's go!

Where Did it Come From?

XML is a subset of a family of 'markup languages'. This family includes HTML, XHTML, MathML and other markup languages, which are being created at a phenomenal rate.

All of these markup languages are derivatives of the Standard Generalised Markup Language, or SGML. To locate the syntax of these standards, have a look at the World Wide Web Consortiums's (W3C) website at www.w3.org. Here you will find not only the current standards, but draft specifications for new markup languages.

If you're serious about learning XML, you should keep checking this resource.

It's quite easy to write an XML document. First of all you must declare the standard that the document will comply with. Then you make a statement. Here's what it might look like:

<?xml version 1.0 ?>

<statement>This is my statement<statement>

The key thing here is that we can see that the document contains a statement, as the content is surrounded by an '**element**' named statement. As you will see later on, elements are themselves surrounded by angle brackets.

We refer to such a document as being **human readable**, because we can read the XML file and make sense of it.

However, the power of XML is such that by creating our own elements (in angle brackets), we can make a document **machine readable**. To get started, let's look at a markup language we are all aware of: **HTML**.

What's HTML got to do with it?

If you have ever designed a web page then you will have encountered HTML. Even if you haven't, you will have almost certainly heard about it. HTML stands for **Hypertext Markup Language**. It is derived from SGML, which was created so that documents could be structured and defined in a standard way.

It's important to see how HTML operates, as it contains everything we have talked about so far - plus some other bits too.

Learning Check: What are elements? What are attributes?

Computers need a standard way of encoding data so that information can be effectively interchanged. The agreement of common terms facilitates that communication, hence the impact of HTTP and HTML on the growth of the web.

ISO 8879 describes SGML, which was defined to allow data and information to be represented, independently of devices or computing platforms. HTML, a language that conforms to SGML, is referred to as a **SGML application**. The SGML standard is extremely comprehensive and considered too complicated for 'general' internet use.

HTML is a simplified version of SGML, designed to allow content to be presented on the web, in a relatively simple fashion. However HTML cannot be extended beyond presentation, so it is limited to producing human readable documents. Unlike HTML, XML is **not** concerned with presentation. It is a standard that permits the **creation** of markup languages. If we can define our own elements, attributes and data types, then we can define the **structure** of a document - XML allows us to do this, and therefore create **machine readable** documents.

Section 2. Understanding Markup Languages

What software do I need?

You don't need anything more than the browser you are probably viewing this tutorial with, and a text editor like Microsoft Notepad (included with Windows) or emacs.

There are tools available to speed up the development of XML applications, but we won't be using them during this tutorial.

Defining the Terms

Throughout this tutorial we shall be referring to a variety of terms. It is important to understand these terms, which are often used, but many are misunderstood.

When we are creating a web page, We use a **markup language** (HTML) to tell the browser how to display the **content** we wish to present. **Content** refers to text, graphics and anything else that is in a digital format.

As mentioned earlier, HTML is the most commonly used markup language. HTML documents contain content and HTML commands (referred to as tags), which determine how the browser will render the document content. If you want to have a look at the HTML markup of a web page, go to **View** and select **Source** in Internet Explorer.

Some HTML

HTML came before XML, and many of the ideas about document construction and presentation are shared between the standards. The latest version of HTML, eXtensible Hypertext Markup Language (XHTML), is an XML language for HTML, ensuring that it conforms to the strict XML standard.

Let's look at a simple HTML document

<HTML>

<HEAD>

<TITLE>Title of Document</TITLE>

</HEAD>

<BODY BGCOLOR="#CCFFFF">

This is the document content.

</BODY>

</HTML>

We write HTML as plain text, using something like Notepad. In fact, we can have a go right now.

Learning Check: Open up a text editor and type in the HTML markup above. Save the file as 'firstpage.htm' and then open it up in your browser. Check the source of the file from within the browser.

HTML Elements

Pages for the web written in HTML are constructed from **elements**, that are defined by **tags**. We can spot tags by looking for the angle brackets (< and >). Take a look at the code in your browser; the first tag is <HTML>. This is referred to as the **root** element as all of the other elements are contained within it. Note that the document ends with </HTML>. This is the **closing** tag.

The containment of tags within the root element is called **nesting**. If you have experience of programming then you will have seen **nested** code, which is a block of instructions contained within other code such as a loop.

In other words, nesting describes a relationship between elements whereby the starting tag of a **parent** element occurs before the starting tag of a **child** element. Similarly the ending tag of the **child** element occurs before the ending tag of the **parent** element.

<parent>

<child>

</child>

</parent>

If we look at 'firstpage.htm', all of the elements are nested within the <HTML> tag. The <TITLE> tag is contained within the <HEAD> tags. Note that the content of the <TITLE> element tag is typed straight in without any formatting. It is the HTML markup that describes the formatting for the browser to interpret.

Attributes

Looking at the <BODY> tag we notice that there are some additional bits of information contained within the angle brackets. BGCOLOR is an attribute that allows us to alter the background colour of the page. It is set using a hexadecimal colour code, which in this case is #CCFFFF (light blue).

Activity: Locate a source on the web where you can access the hexadecimal colour codes for use in web pages. Bookmark the page you have found for future reference.

Getting onto the Web

So, we know that we have to markup our content with HTML element tags, and save that file as plain text. We can use either '**.htm**' or '**.html**' as our file extensions. Now we have to put this file onto the public area of a web server.

When a request for a web page is made over the Internet, the web server sends a copy of the text file to the client (browser). The browser then interprets the HTML markup and renders the content of the file accordingly.

Activity: We shall now create a HTML document. Follow the instructions below step-by-step.

1. You will need a text editor and a web browser installed on your machine. Open the text editor and save the document as 'secondpage.htm'.

2. Type <HTML> on the first line. Press the Carriage Return key on the keyboard to start a new line and type <HEAD>.

3. Give the document a title - how will you markup the title?

4. Set the background colour of the document to white, with black text and blue links.

5. Enter some content. Use boldtype and italics to emphasise important points.

6. Check the nesting of your element tags, and save the final version of your file.

7. Open the file in your browser and admire your results! If you are feeling adventurous, publish it to some webspace and view it over the internet.

Document Type Definition

There are three Document Type Definitions (DTDs) available for use with HTML documents. But what is a DTD?

We have seen that HTML is a SGML application, that is, it conforms to the rules set out by the SGML standard. Within that standard, we must define which **elements** and **attributes** are allowed. The DTD allows us to do this.

The Strict DTD contains all elements and attributes that have **not been deprecated**, or made obsolete.

The Transitional DTD includes everything in the Strict DTD, plus all deprecated elements and attributes.

The Frameset DTD has everything in the Transitional DTD as well as attributes and element required to support frames.

Formally, we should declare which DTD a HTML document will use. This will assist the browser when rendering the page.

<!DOCTYPE HTML

PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN"

"http://www.w3.org/TR/html4/frameset.dtd"

<HTML>

<HEAD>

...

The sample HTML above would tell the browser that the document is based on the **Frameset DTD** and therefore frames will be used.

Section 3. Exercise 1

Research Activity

Using a search engine such as 'Google', find ten examples of XML based markup languages. Write a HTML file to display the name of the language, plus a short description. Save this file as 'markups.html'.

After you have presented the content, can you think of an improved way of presenting the tabular data? Which of the HTML element tags would you use?

Section 4. XHTML

What's the difference between HTML and XHTML?

The eXtensible Hypertext Markup Language is the new version of HTML. It is completely XML compliant, but what does that really mean?

The HTML standard is actually a recommendation, and to this end you have probably witnessed the different ways in which competing browser manufacturers choose to interpret the element tags. It is not uncommon for a HTML page to be rendered differently, and web page designers have for a long time tested their creations in different browsers to identify any deviations from their original intentions.

Also, the browser manufacturers have made it 'easier' for individuals to create HTML pages by allowing us to 'forget' to close some of the element tags. This would break the 'nesting' rule, but allowing it prevents novice users from getting bogged down in debugging. The downside is that it permits the production of 'sloppy' HTML code, making it un-friendly for machine reading purposes. XML is strict, and will not allow such practices.

XHTML uses the strictness of XML to tighten up the formality of HTML document preparation. A document that conforms is regarded as being **well formed**.

The Rules

* All documents must include a DOCTYPE statement
* The root element of the document must be <HTML>.
* Elements and attributes must be lowercase.
* Attribute values must be enclosed in quotation marks and not minimised.
* Leading and trailing spaces in attribute values will be stripped.
* Only the id attribute can be used to identify an element uniquely.
* Non-empty elements must be terminated or have an ending tag.
* Nested elements must be properly nested, not overlapping.
* SCRIPT and STYLE elements must be marked as CDATA areas.

Wow, information overload - let's break it down.

Declaring a DOCTYPE

We know already that the DOCTYPE declaration can be left off a HTML document. All XHTML documents have a **mandatory** requirement for this declaration, otherwise it will not comply with the rules for **well formed**ness.

<!DOCTYPE HTML

PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"

"http://www.w3.org/TR/xhtml1/frameset.dtd"

This is the declaration for a **Frameset DTD**, and the **Transitional DTD** is:

<!DOCTYPE HTML

PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"

"http://www.w3.org/TR/xhtml1/transitional.dtd"

For a **Strict DTD** declaration it is:

<!DOCTYPE HTML

PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"

"http://www.w3.org/TR/xhtml1/strict.dtd"

Note that the URL for the DTD has been amended, as it is a different file, and the XHTML version has been inserted.

Root Element

To prevent overlapping elements, and therefore force proper **nesting**, XML requires a root element. In XHTML this is <html> and is placed after the DOCTYPE declaration.

Also there must be a reference to an XML namespace that the document uses. We haven't covered them yet, but this is what one looks like:

<html xmlns="http://www.w3.org/1999/xhtml">

Upper or Lower Case?

XML is case sensitive, so it matters how you type your elements and attributes. With HTML you get away with mixing cases, but in XML you can't. The same goes for XHTML.

Here is an example of a HTML document:

```
<HTML>

<HEAD>

<TITLE>Title of Document<HEAD>

</HEAD>

<BODY BGCOLOR="#CCFFFF">

This is the document content.

</BODY>

</HTML>
```

...and here is the XHTML version, after applying the XML well formed rules:

```
<!DOCTYPE html

PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"

"http://www.w3.org/TR/xhtml1/transitional.dtd"

<html xmlns="http://www.w3.org/1999/xhtml">

<html>

<head>

<title>Title of Document</title>

</head>

<body bgcolor="#CCFFFF">

This is the document content.

</body>

</html>
```

The attribute values are exempt from the case sensitivity rule, as well as filenames in URLs.

No Minimised Attributes

What do we mean by **minimised** attributes?

With HTML, attributes appear in a name-value pair, unless they are **minimised**. This looks like:

<INPUT TYPE="radio" NAME="CCType" VALUE="Visa">

<INPUT TYPE="radio" NAME="CCType" VALUE="Mastercard">

The data returned from these input elements will consist of "CCType" and either "Visa" or "Mastercard", thus constituting the name-value pair.

However the **SELECTED** attribute has been minimised below - what this means is that the minimised attribute works **without a value**.

<INPUT TYPE="radio" NAME="CCType" VALUE="Visa" SELECTED>

<INPUT TYPE="radio" NAME="CCType" VALUE="Mastercard">

So when the web page is displayed, the HTML above will show the first value already selected.

However, XML demands that minimised attributes are written as a name-value pair, and so XHTML follows suit. Our 'SELECTED' attribute now must become:

<input type="radio" name="CCType" value="Visa" selected="selected">

No White Spaces

When XML files are parsed, extra white space is stripped from between words. 'Extra' means more than one character spacing. This also means that extra white space in an attribute value will be reduced to one space, potentially causing bugs.

The id Attribute

In HTML, each element has a 'NAME' attribute that allows each element to independently identified. XHTML uses the **id** attribute to identify elements. However like the rest of the XML standard, its use is more tightly defined than with HTML. In particular, only **one** instance of an id can be used **per document**.

Non-empty Elements

HTML allows elements to be started and not terminated - if you miss a paragraph end tag, <P> then the paragraph will still be rendered. With XHTML the end tag must always be included, **unless the element does not require any attributes.**. For example:

 is permissible for the HTML line-break (
) in XHTML.

For an element that has no content, both starting and ending tags must be used as follows:

Nesting Elements

Improper nesting, or **overlapping** is prohibited in XHTML. Whilst the following may be acceptable in HTML:

<H1><I>Using italics in a heading</H1></I>

...you will need to write it this way for a well formed XHTML document:

<h1>Using italics in a heading</h1>

CDATA

When we write client-side validation scripts, or dynamic content in JavaScript or VBScript, we have to use special characters to make sure that the code is not interpreted as HTML.

XHTML has a specific declaration for such instances:

<script language="Javascript">

<![CDATA[

function clickme() {

alert("Hello")

}

]]>

</script>

Section 5. Exercise 2

Activity

Using the 'secondpage.html' file you created during this tutorial, create a XHTML compliant version and call it 'xhtmlpage.xhtml'.

Now do the same with the 'markups.html' file.

Practice What You Preach

This tutorial was written entirely in XML, using XSLT stylesheets and transformations to convert an XML file into a number of HTML pages, a zip file, JPEG heading graphics, and two PDF files. This ability to generate multiple text and binary formats from a single source file illustrates the power and flexibility of XML.