IBM IT Education Services

**Session ID: E14**

**Speaker : Janice Winchell**
**MICOM Inc**

# *XML ... Pretty HOT stuff*

November 10 - 12, 2003 | Hilton, Las Vegas, NV



RETURN TO INDEX

RETURN TO INDEX

# Introduction

# to

# XML

## XML - pretty ***HOT*** stuff

Introduction to XML

- XHTML as well

HTML can do a lot, but XML can do more

What are the parts of an XML document?

What is the syntax (grammar) that XML expects?

More things to understand

- Document Type Definitions (DTD)
- XML namespaces
- XML schemas
- XSL - extensible style sheets
- XML Linking
- Document Object Model (DOM)

Creating XML documents

- Elements, Attributes, Entites

Where to find out more information

- Additional resources

***..OK....fasten your seat belts - XML is going to be an exciting ride!***

**The most visible definition of XML comes from the XML Specification V1 of February 1998**

**"The eXtensible Markup Language (XML) is a subset of SGML (Standard Generalized Markup Language) that is completely described in this document. Its goal is to enable generic SGML to be served, received and processed on the Web in the way that is now possible with HTML. XML has been designed for ease of implementation and for interoperability with both SGML and HTML"**

**Or, you can say XML *is* SGML**

**"XML is an application profile or restricted form of SGML, the Standard Generalized Markup Language (ISO 8879). By construction, XML documents are conforming SGML documents.**

## Design goals for XML

- XML - straightforward and usable over the internet
  - Not just over the internet, but also within the enterprise
- XML - support wide variety of applications
  - Beyond tools that live on Web servers
- XML - compatible with SGML
  - Have to keep enough SGML for interoperability
- Make it easy to process XML documents
- Keep optional features to absolute minimum, ideally zero
  - SGML options make for flexibility and power, but also make for complication
- XML documents - need to be human-legible and reasonably clear
  - Allows use of text editors
- XML design - prepared quickly
  - Didn't want to spend TOO long in development creating another browser war
- Design of XML - formal and concise
  - Makes language programmer friendly
- XML documents - easy to create
  - Allows wide variety of creation tools for many varied users
- Terseness in XML markup minimally important
  - Avoid complexities of SGML
  - Choose clarity over cleverness

## XML and its relations

.....XML is not “one technology” - instead a *FAMILY of related standards*

- **eXtensible Style Language (XSL)**
  - Similar to Cascading Style Sheets (CSS) working with HTML
- **eXtensible Hypertext Markup Language (XHTML)**
  - Deliver on different media types (palm, etc)
- **XML Linking Language**
  - **Xpath**
    - Perform actual addressing of parts rather than entire XML document
  - **Xlink**
    - Use XML syntax to define relationship between 2 or more data objects or portions of objects (not just entire document)
  - **Xpointer**
    - Builds on XPATH for internal addressing using XML markup to link
- **XML Namespaces**
  - Allow assigning unique name to document constructs (avoid duplicate names)
- **XML Schemas**
  - Allow constrains and rules
- **Document Object Model (DOM)**
  - Allow language neutral application interface to manipulate HTML and XML data
- **Scalable Vector Graphics (SVG)**
  - XML application for rendering 2-dimensional graphic objects

## Why use XML?

- **Structured data**
  - Can extract just the information necessary
- **Data exchange**
  - Allows exchange of database contents
  - Can send information in a structure that can be parsed (you define the tags)
- **XML complements HTML**
  - XML data can be used in HTML page
- **Self-describing**
  - No prior knowledge of an application is needed
- **Search engines**
  - Search relevancy increases due to contextual information in XML document
- **Updates**
  - Document Object Model built into XML allows access and update of individual elements (not required to update entire site page by page)
- **User-selected view of data**
  - Different users, different media can access different information or present information in different ways

## Why use XML?

- **Can replace proprietary tagging systems with platform independent meaningful tags**
  - Applications can now parse this information

```
<!--Price List for veggies -->
<d1>
  <!-- Veggie -->
  <dt> Broccoli</dt>
    <!-- Price -->
    <dd>$1</dd>
  <!-- Veggie -->
  <dt>Cauliflower</dt>
    <!-- Price -->
     <dd>$2</dd>
</d1>

now can look like:

<VeggiePrice>
  <Veggie>Broccoli</Veggie>
    <Price>$1</Price>
  <Veggie>Cauliflower</Veggie>
    <Price>$2</Price>
</VeggiePrice>
```

## Tools for XML - You can build it yourself “by hand” with text editor

Not a bad idea if you are just starting

1. and not a bad way to get the “practice” and understanding

- Make the XML Declaration (not required but recommended to have one)
  - <?xml version=“1.0” standalone=“no” encoding=“UTF-8”?>

- Create root element
  - You pick the element “name”
- Create XML Code
  - Tags have start and end components
  - Tags are nested correctly
  - Attribute values must have quote bounds (either “ or ‘)
  - Empty tags must be correctly formatted (use a / before the closing >)
- Check the coding (these are just a couple of places)
  - IBM’s can be found by going to

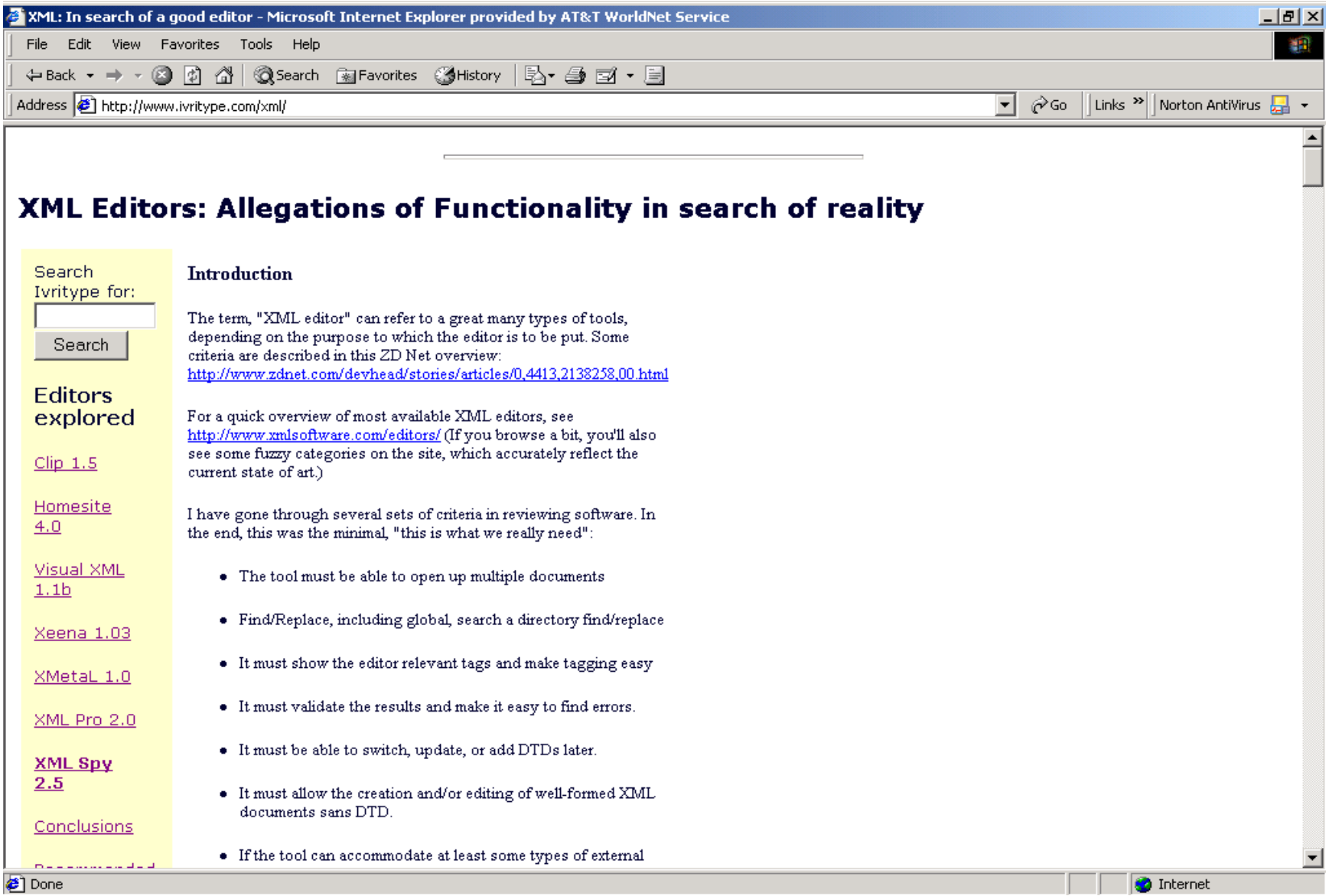
    www.ibm.com, searching on XML Validator and picking one that starts out so:

    http://www- .ibm.com/software/webservers/appserv/doc/v30/ae/web/apidocs/com.ibm….
    - the actual url is way long!
  - Microsoft’s XML Validation page
    - http://msdn.microsoft.com/downloads/samples/internet/xml/xml_validator

# Tools for XML

# Tools for XML



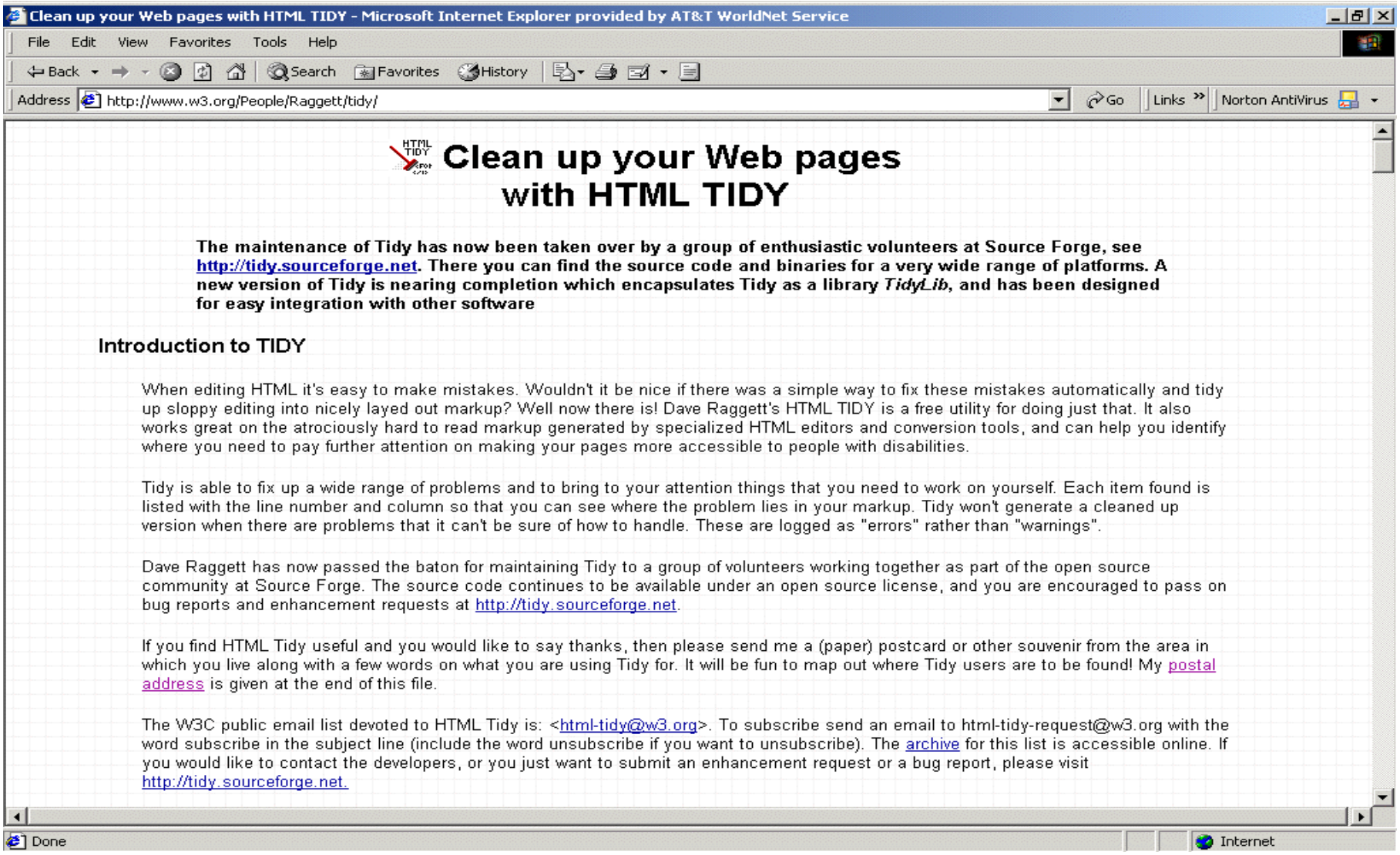

  

## XML - pretty ***HOT*** stuff

- XHTML - intended to provide easy transition between HTML and XML
  - Extensible Hypertext Markup Language
    - HTML like markup language
    - Users can extend the language
    - XHTML is defined as an XML DTD
      - HTML is defined as an SGML DTD
    - Designed to address the issue of how to re-use existing HTML
    - Retain compatibility with existing browsers
    - Extend with new features for XML-enabled
    - Allow automatic validation of Web pages
      - eliminate common coding errors
      - retain compatibility with “legacy” browsers
  - One of many XML languages
  - Big opportunity for e-commerce transactions and information exchange using XML vocabularies
  - Render document appropriately for different media and browsers

- XHTML - transition between HTML and XML



HTML revisited

(why do I need XML anyhow?)

- HTML
  - *Presentation* language
  - Tags define how to display text
  - HTML makes web stuff look “pretty”
  - HTML describes *HOW TO DISPLAY*
  - HTML does not *easily* describe page CONTENT

  NOT meant for
  - *Tight control* of document display
  - *Flexibility* for different but specific types of info
  - Rendering information in *variety of media and formats*
  - *Publishing* single set of information *across this variety*
  - Defining *complex linking* relationships

- HTML
  - Lacks fine controls
    - Cannot specify display size of document
    - Cannot control window size of browser windows
  - Display output varies depending on browser
    - Internet Explorer
    - Netscape Navigator
    - Mosaic
    - HotJava
    - Others

New tags and style sheets help with HTML

- IS EASY, QUICK, and CHEAP TO USE
  - HTML can be created with text editor and a little knowledge
- Is very forgiving of “errors”

## XML - pretty ***HOT*** stuff

- HTML and XML
  - Both derive from SGML
  - Both use tags
  - Both use attributes
- HTML
  - says HOW text is displayed in browser window
- XML
  - says what each word means
  - Formal and precise
  - Flexible and growing
  - Structured
- XHTML
  - Reformulated HTML 4.0
  - Application of XML

- XML
  - Developed under WWW Consortium umbrella
    - That would be the W3C
  - Objective was to bring SGML (standard generalized markup language) to the web
    - SGML is a language designed to talk about other languages
    - Parent to HTML & XML (and many other markup languages)
  - SGML - the “kitchen sink” (and still under construction)
  - Has EVERYTHING anyone would EVER need for ANY markup language situation
    - SGML is a metalanguage
    - Extra bells and whistles because it needs to be EVERYTHING to EVERYONE
    - Makes implementation difficult and language slow
  - XML - the *trim and slim* implementation of SGML
    - Designed from the outset for the web

- XML document
  - Well-formed (this is a basic XML rule)
    - Must have start & end tags for *every element*
    - One and *ONLY ONE root element*
    - Empty elements must be formatted correctly
    - *CASE* of start and end tags can be EITHER UPPERCASE or lowercase, but MUST MATCH
    - Elements must *NEST* correctly
    - Attribute values must *ALWAYS BE IN 'QUOTES'*
      - Single or double, just make them the SAME
  - Valid (*optional to validate*)
    - Contains **<!DOCTYPE declaration**
      - But unless it is checked it is not considered VALID
    - (Well-formed) AND VALIDATED against a DTD or SCHEMA
    - XML parser has determined document CONFORMS TO RULES
  - Can have Well-formed only or WELL-FORMED AND VALIDATED
    - There is an entire class of "non-validating processors" which *do not USE the DTD (Document Type Definition)*

## HTML vs XML - a simple example

```
<html>
<head>
<title>XML Sites</title>
</head>
<body>
<h1>Web sites on XML </h1>
<p>Here is a list of some <em><font color=red>XML Web sites</em></font>
<ul>
<li>http://www.xml.com
<li>http://www.xml.org
<li>http://alphaworks.ibm.com/xml
</ul>
<br>
>img src="file.gif' alt="image name">
</body>
</html>
```

This is the HTML code you are familiar with

- HTML vs XML - a few changes make a correct XML example that looks VERY FAMILIAR! (after you add the ending tags and fix the order - required by XML)

```
<html>
<head>
<title>XML Sites</title>
</head>
<body>
<h1>Web sites on XML </h1>
<p>Here is a list of some <em><font color="red">XML Web sites</font></em></p>
<ul>
<li>http://www.xml.com</li>
<li>http://www.xml.org</li>
<li>http://alphaworks.ibm.com/xml</li>
</ul>
<br />
>img src="file.gif" alt="image name" />
</body>
</html>
```

- XML requirements

What I had to change from prior HTML code to conform to XML rules

- Elements are CASE SENSITIVE (must match)
- All elements MUST have *start* tag and *end* tag
  - Added the ending paragraph tag </p>
- Corrected the order of the closing tags for the </font> (along with adding the </p>)
  - Must close in EXACT REVERSE ORDER
  - Close first what you opened last
- All attribute values must be in quotes (single or double just make them match)
  - Changed
    - <font color=red> to
    - <font color="red">
- Singleton or "empty" tags must be closed

  Requires space followed by slash or /tag)
  - <br> becomes <br />
  - <img src="file.gif" alt="image name"> becomes
    <img src**="file.gif" alt="image name" />**

XML documents

the “parts”

- **XML document - the parts**
  - XML declaration
    - Typically first item in document
  - Document Type Definition (DTD)
    - Required if document is to be validated via validating parser - which is what is also required to have a “valid” document
    - May be internal to document
    - May reference a URL
    - May be a combination of internal + URL
  - Document Element
    - Single element required
      - Root element or Document element
      - This is the top-level tag - you choose the name
      - All tags and contents live within this tag
        - Under HTML this was the <HTML> .... </HTML> tag combination
  - Document Content
    - Elements
    - Attributes
    - Comments
    - Processing Instructions

- **XML Declaration or prologue**
  - **identifies the language**
  - **Provides version number**
    - **Very similar to DOCTYPE element in HTML which looks like:**

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
```

    - **where in XML you would see:**

```
<?xml version="1.0"?>
```

    - **For XML the ? indicates a construct called a *processing instruction (or pi)***
  - HTML typically uses uppercase (but you can write it anyway you want Html HTml html, etc since folding occurs)
  - XML cares about case (this also makes XML global)
  - Simplest doctype declaration is one that references URL

- **XML Document Element**
  - Must have single element containing all other elements
  - This is the *ROOT ELEMENT* or *DOCUMENT ELEMENT*
    - Note that in HTML this is <HTML> since all other content is inside the begin/end tags <HTML> </HTML>
  - In XML you CANNOT use XML as root (XML is reserved)
    - Here is an example where ***MEMO*** is the document element!

```
<?xml version="1.0"?.
<!DOCTYPE MEMO "http://www.jmw.com/memo.dtd"
<MEMO>
<TO>William Winchell</TO>
<CC>Donald McCord</CC>
<FROM>Janice Winchell</FROM>
<DATE>10/11/2003</DATE>
<RE>Upcoming travel plans</RE>
<BODY>The next planned trip will be in November to the
IBM Technical Conference in Las Vegas from November 10-12, 2003.
Please have all your travel documents ready, and your presentation
material ready to ship by October 3, 2003. </BODY>
</MEMO>
```

- **Document Content**
  - Elements
    - Primary components of a document
    - Bound by start and end tags
    - May have CDATA(character not markup data) describing element
    - May be “empty”
      - That is *no content* between open and close tag
      - *May* have attributes

      <img src=“mymgb.jpg”></img> is just as empty as

      <br /> or <br></br>
  - Attributes
    - Property of an element
    - information *WITHIN* an element’s start tag
    - <car type = “MGB”></car>
    - Composed as *name=value* pair
      - *name* is name of attribute (type)
      - *value* is the value(MGB)

## XML Comments

Syntactically equivalent to HTML and SGML comment

- Begin with markup declaration open delimiter **<!**
- and then a comment open delimiter **--**
- So, you end up with <!--
- Close comment with a close delimiter and a markup declaration close delimiter, or -->
- Any character data may be contained in the comment
  - *Not the > character*
  - *Not the double hyphens* --
- Comments are important for later reading (just as in any programming language)
- Comments may appear anywhere
  - Must be outside other markup
  - Not immediately prior to eh XML declaration

**<?xml version="1.0"?>**

  - Here is a valid comment:

<!-- Put your comment here -->

  - Note that SGML allows "inline comments" - XML does NOT

- **Processing Instructions**
  - Provide means to send instructions to
    - Computer Program
    - Application
  - Bounded by <? and ?>
  - The information is passed through to the application
  - An XML parser *may or may not pass on comments*
  - A conforming parser ***MUST*** pass on processing instructions

**<?noisemaker noise='cello.wav' ?>**

- The application is noisemaker
- The attribute noise and its corresponding value cello.wav
- The noisemaker application should now make the noise contained within the cello.wav sound file

- Another typical example involves style sheets in an XML doc

**<?xml:stylesheet type="text/xsl"?>**

- or how about a familiar one for javascript:

**<?javascript (javascript stuff) ?>** or

**<?Tcl (tcl stuff) ?>**

XML

the grammar rules

- **XML employs the Extended Backus-Naur Form (EBNF) (with added commentary)**

  **Note:John Backus and Peter Naur described formal notation for describing syntax of ALGOL60 computer language - still a pretty good model today and what the W3C group used for XML (plus comments)**

  - Basic notation:
    - symbol ::= expression
      - where ::= represents “is defined as”
      - Example: new symbol vowels ::= [aeiou]
    - Rules can refer to range of sequential characters
      - Example: FirstFive ::= [a-e] representing the first 5 characters of the English alphabet
    - Remember that ***case counts*** so maybe we should say
      - vowel ::=[aeiouAEIOU]

      if we want both upper and lower case to apply

- **EBNF**

Can also handle complex expressions

such as “one or more of these options:

- S ::= (#x20 | #x9 | #xD | #xA) +
  - S (represents white space)
  - #x20 is hex 20 (ASCII 32
  - #x9 is the tab
  - #xD is the newline
  - #xA is the linefeed
    - The | is representative of an “or”
    - The + is “one or more”

So this rule reads:

White space is defined as one or more of the space, tab, newline, or linefeed characters

- **EBNF**

  Can also do One But Not The Other and Either/Or

  - AcceptableGrade ::= A - B becomes the rule
    - An acceptable grade is defined as the character A but NOT the character B
  - AcceptableGrade ::= A | B
    - An acceptable grade can be EITHER A OR B

  Excluding Characters from a set:

    - consonants ::= ( [^aeiou] | alphabet)
      - Consonants are defined as the alphabet, except the letters a, e, i, o, u
      - The pipe character, |, separates the subject of the expression (alphabet) from the exclusion set

  - XML grammar rules are written in EBNF, but as you can see this language is fairly TERSE

Document Type Definition

DTD

- **Document Type Definition (DTD)**
  - Sets forth the rules for what makes the document VALID
  - Can be written “internally” right inside doctype definition
  - Can be referenced via URL
  - Might have combination of the 2!
  - Similar to style sheet in XML
  - Identifies the grammar expected
    - Identifies the elements
    - Tells how to use element
  - Builds the structure and nesting
    - Identifies the hierarchy

Good practice to declare everything before using

Makes it easier to find/dissect the information

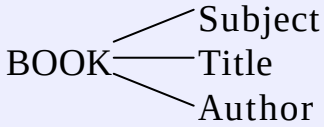
# Rules of the DTD road

| TERM | Example | What it does |
|---|---|---|
| **XML declaration** | <?xml version = “1.0” ?> | Identifies version of XML to use |
| **Document type declaration** | <!DOCTYPE Root-Element SYSTEM “Root-Element.dtd”> | Identifies where to find the dtd (presuming it is external but in the same directory (URI)) |
| **Element type declaration** | <!ELEMENT Name (#PCDATA)> | Defines element type |
| **Attribute list declaration** | <!ATTLIST Element-Name Name Data-Type Default> | Defines name, data type and a default value for each attribute |
| **Entity declaration** | <!ENTITY Entity-Name “text”> | Defines information that can be “called” using entity name |
| **Notation declaration** | <!NOTATION Name System ‘externalID”> | Association name to find the interpreter for the notation |

- **DTD Example (inline)**
  - Comments are always appreciated
    - Never nest comment in another tag
    - Never use a hyphen in your comment
  - ELEMENT defines the name (and more)
  - ENTITY defines alias for block of text or external content

```
<!DOCTYPE Book-Review [
<!ELEMENT Book-Review (Book*)>
<!-- Only a single subject, single title, and single author are expected/allowed -->
<!--Data is parsed character data (#PCDATA)   (same as text) -->
<!--Book has 3 subordinate items, Subject, Title, and Author) -->
<!ELEMENT Book (Subject, Title, Author)>
<!ELEMENT Subject (#PCDATA)>
<!ELEMENT Title (#PCDATA)>
<!ELEMENT Author (#PCDATA).
<!ENTITY Publisher “Winchell Books”>
]>
```



- **DTD**
  - Book-Review is the root element
  - Element type declarations name the elements and identify children
    - <!ELEMENT Name ANY> can have data or markup info
    - <!ELEMENT Name EMPTY> must contain no content
    - <!ELEMENT Name (#PCDATA( | ChildName)>
    - <!ELEMENT Name (Child1, Child2)>

```
<?xml version = '1.0' encoding='UTF-8' standalone='yes' ?>
<!DOCTYPE Book [
<!ELEMENT Book-Review (Book*)>
<!--  Subject is optional or repeatable, multiple authors allowed -->
<!-- Order is as specified, subject then title then author -->
<!ELEMENT Book (Subject*, Title, Author*)>
<!ELEMENT Subject (#PCDATA)>
<!ELEMENT Title (#PCDATA)>
<!ELEMENT Author (#PCDATA)>
<!ENTITY Publisher "Winchell Books">
]>
```

- **DTD**
  - Entities
    - Provide “shorthand”
    - May be a placeholder
    - Works like parameter substitution
    - Two “high-level” types
      - General
      - Parameter
    - Four subsets
      - Internal entities
        - References to definitions entirely within DTD
      - External
        - References to definitions OUTSIDE the document DTD
      - Parsed
        - XML processor can and will parse
      - Unparsed
        - Handed off to another application

- **General Entities**
  - Created in DTD
  - Used in XML document
- **Parameter Entities**
  - Created AND USED in the DTD

- **Predefined or "built-in" Entities (no need to declare these)**

<  <

>  >

'  '

&quote;  "

&  &

- Benefits of using entities beyond just parameterization
  - Create reusable - can create separate pieces and use external entities to "include" them

- **DTD how to get started**
  - **Step 1 is to analyze your data!!!**
    - **Create data structure (data and children)**
  - Step 2 - Define element names (this is your data dictionary)
  - Consider putting information into parameter ENTITY
    - Allows easier changing later and makes DTD global ready
  - ENTITY makes it easier to translate entity declarations without changing the underlying DTD
    - Publisher is the English word
    - Verleger is the German word for publisher

```
<!-- Element names -->
<!ENTITY  % Doctype          “doctype”
<!ENTITY  % Title            “title”>
<!ENTITY  % Author           “author”>
<!ENTITY  % Date             “date”>
<!ENTITY  % Publisher        “publisher”>
<!--Document root -->
<!ELEMENT  %Publisher;
```

## ENTITY markup rules

- Must be declared in DTD or schema
  - If document not being validated must create enough of a DTD within your XML to at least declare the entity you want to use (excepting the pre-defined ones)
- *General entity* reference must have &myEntity; format when used in doc
- Name must ***begin*** with letter or underscore
  - Can contain letters, underscores, whole numbers, colons, periods and/or dashes
- Declaration cannot have markup the begins in entity declaration and ends outside
- *Parameter entity* must have preceding percent sign (%) with white space before and after, and then reference with %name no space

```
<!-- Element names -->
<!ENTITY  % Doctype              “doctype”
<!ENTITY  % Title                “title”>
<!ENTITY  % Author               “author”>
<!ENTITY  % Date                 “date”>
<!ENTITY  % Publisher            “publisher”>
<!--Document root -->
...
<!ELEMENT  %Publisher;
```

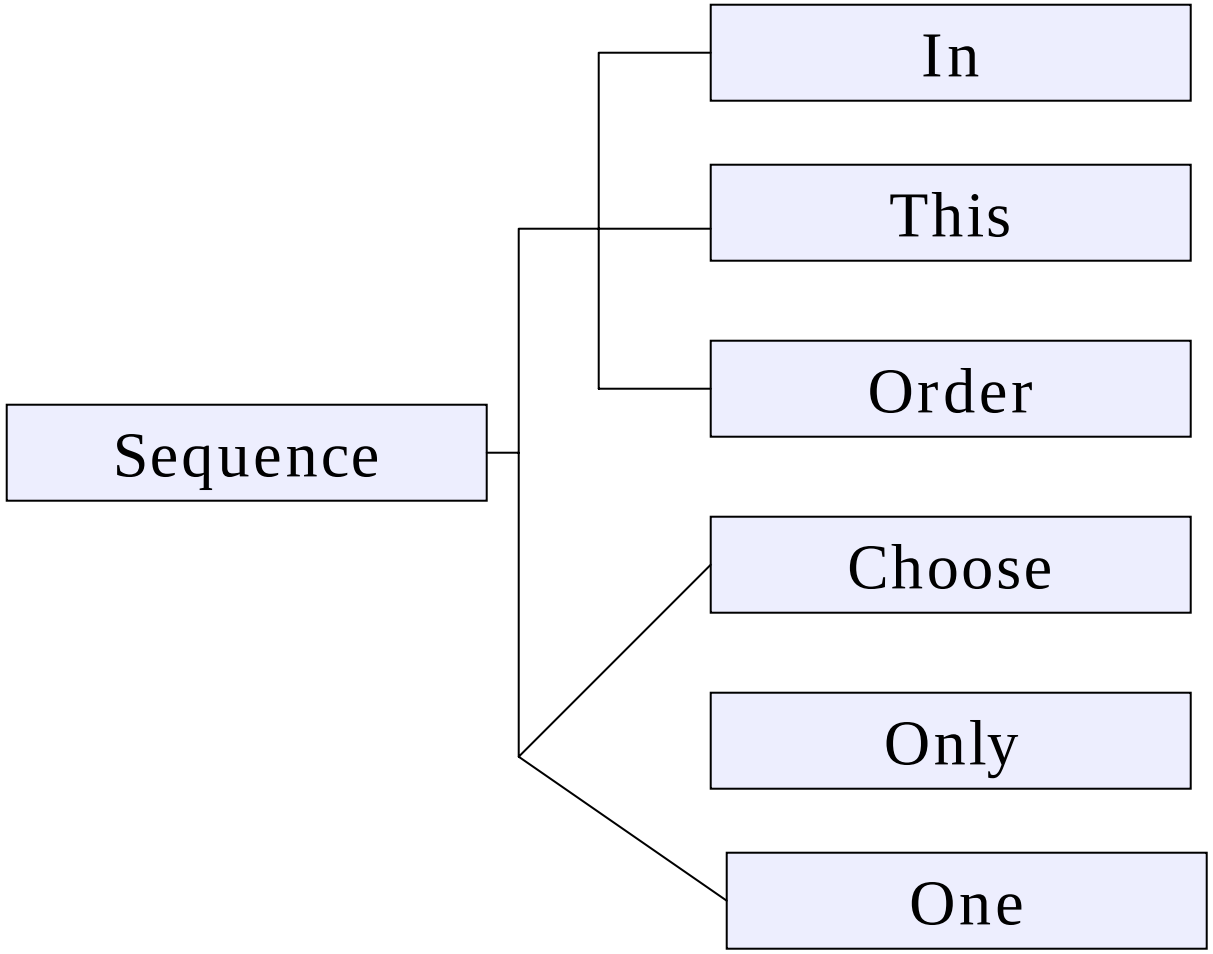
  

## XML Entity coding sample snippit from XML site

```
<?xml version='1.0' encoding='UTF-8'?>
<?xml-stylesheet type="text/xsl" href="xmlspec.xsl"?>
<!DOCTYPE spec PUBLIC "-//W3C//DTD Specification V2.1//EN" "xmlspec-v21.dtd" [
<!ENTITY year "2001">
<!ENTITY month "June">
<!ENTITY MM "06">
<!ENTITY day "27">
<!ENTITY DD "27">
<!ENTITY internalXMLBase "http://www.w3.org/XML/Group/&year;/&MM;/REC-xmlbase-&year;&MM;&DD;">
<!ENTITY externalXMLBase "http://www.w3.org/TR/&year;/REC-xmlbase-&year;&MM;&DD;">
<!ENTITY XMLBase "&externalXMLBase;">
<!ENTITY LatestXMLBase "http://www.w3.org/TR/xmlbase">
]>
<spec w3c-doctype="rec" status="final">
<header>
 <title>XML Base</title>
 <w3c-designation>xml-base-&year;&MM;&DD;</w3c-designation>
 <w3c-doctype>W3C Recommendation</w3c-doctype>
 <pubdate><day>&day;</day><month>&month;</month><year>&year;</year></pubdate>
 <publoc>
  <loc href="&XMLBase;/">&XMLBase;/</loc> (available in
  <loc role="available-format" href="&XMLBase;/Overview.html">HTML</loc>,
  <loc role="available-format" href="&XMLBase;/Overview.xml">XML</loc>)
 </publoc>
 <prevlocs><loc href="http://www.w3.org/TR/2000/PR-xmlbase-20001220/">http://www.w3.org/TR/2000/PR-xmlbase-20001220
 </loc>
<!--   <loc href="http://www.w3.org/TR/2000/CR-xmlbase-20000908">http://www.w3.org/TR/2000/CR-xmlbase-20000908</loc>
  <loc href="http://www.w3.org/TR/2000/WD-xmlbase-20000607">http://www.w3.org/TR/2000/WD-xmlbase-20000607</loc>
  <loc href="http://www.w3.org/TR/2000/WD-xmlbase-20000221">http://www.w3.org/TR/2000/WD-xmlbase-20000221</loc>
  <loc href="http://www.w3.org/TR/1999/WD-xmlbase-19991220">http://www.w3.org/TR/1999/WD-xmlbase-19991220</loc>-->
 </prevlocs>
 <latestloc>
  <loc href="&LatestXMLBase;/">&LatestXMLBase;/</loc>
 </latestloc>.....
```

- **Structure charts help define data and hierarchy**




This example is representative of structure charts used for XML design by Microstar Software Ltd.

- **Check whether your industry already has a DTD- here are a few that already exist (keep checking)**

| Reference DTD | Use or industry |
|---|---|
| TCIF | Telecommunication Industry Forum Standard |
| SIF | Schools Interoperability Framework |
| HL7 | Health industry for charting and diagnosis |
| TEI | Text Encoding Initiative - book oriented |
| HTML | Web publishing |
| ISO 12083 | Electronic Manuscript Standard for journals, articles, etc |
| SAE J2008 | Spec for documentation of Vehicle Service information |
| DocBook | Specs for Software Documentation |
| ATA Spec 2100 | Airframe/Engine documentation spec |
| MIL38784B (CALS) | Military Specification Technical Documentation |

Namespaces

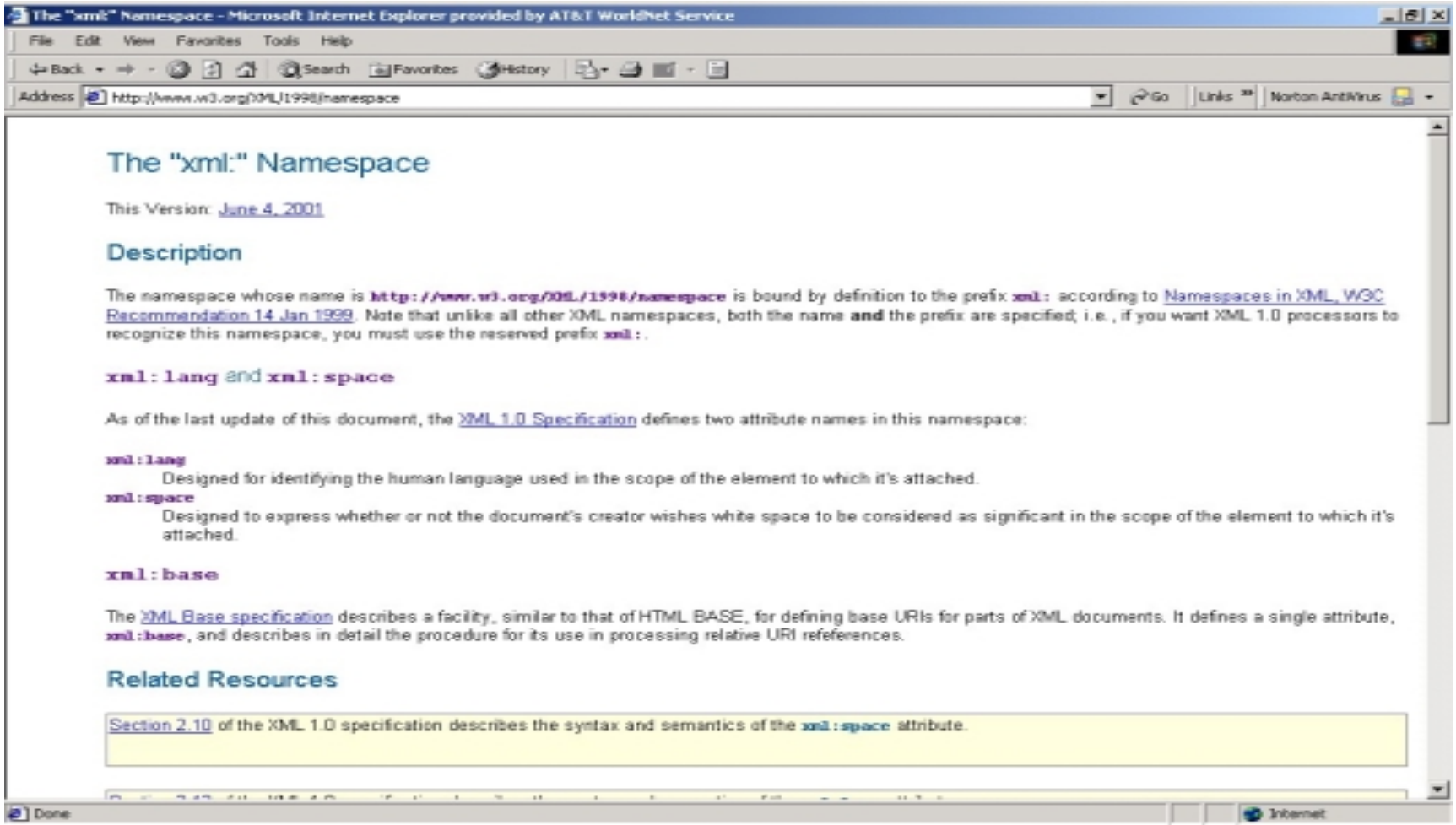
  

- **Why do we need namespaces?**
  - With flexibility comes conflict
    - Documents can be created using the SAME NAMES
      - RECORD might mean a music element OR it might mean a person's vehicle data, or a medical piece
  - Namespaces allow collision avoidance
    - Can use multiple DTD's in the same document and ensure unique name
      - Within a DTD a name MUST be unique
  - Technical definition of namespace from the XML Recommendation:

    "An XML *Namespace* is a collection of names, identified by a URI reference, which are used in XML documents as element types and attributes"

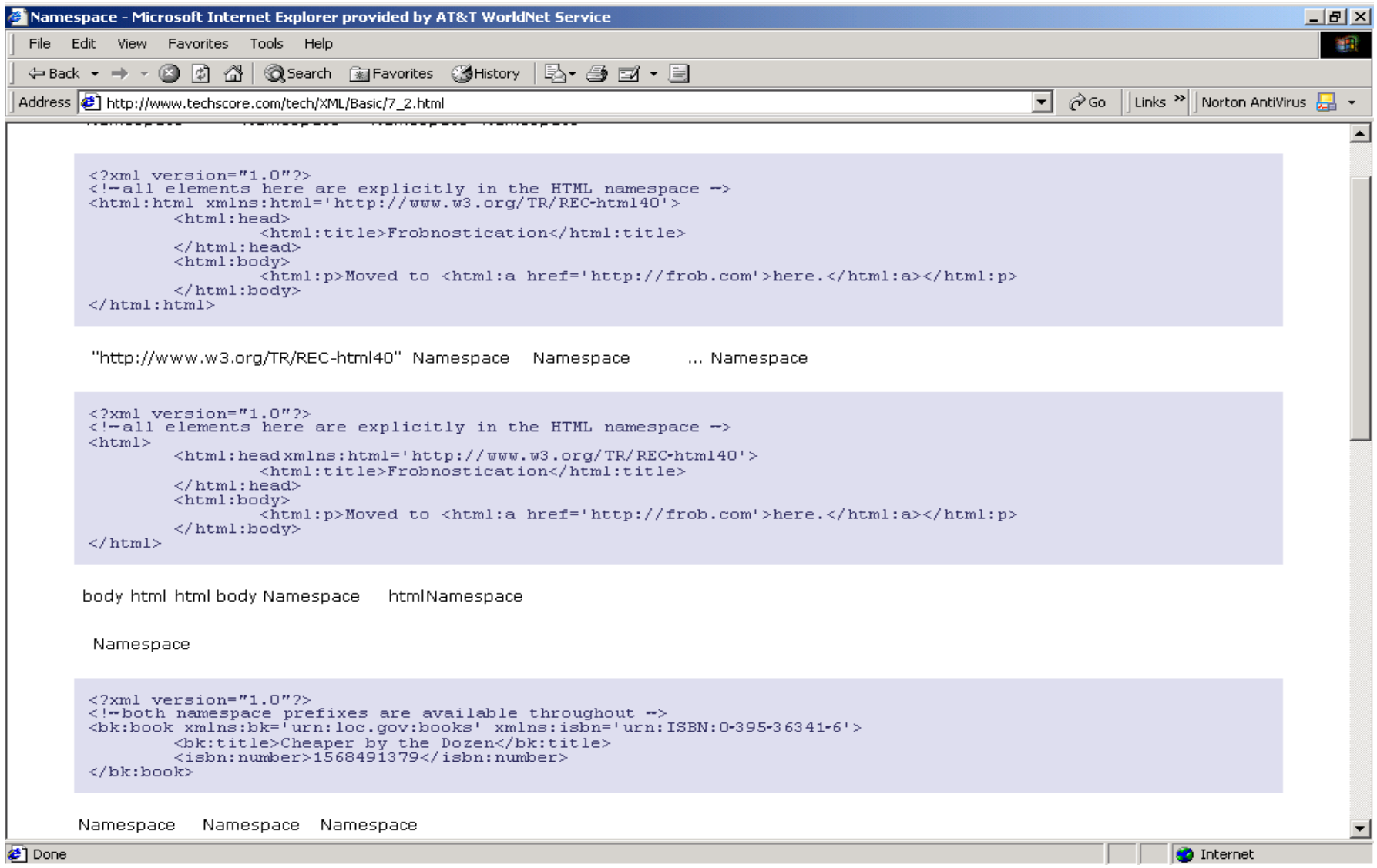
  

- **Some namespaces already exist**
  - XML itself:
    - xmlns:xml="http://w3org/XML/1998/namespace"



- **Here are some namespace coding examples**



- **Declaring a namespace**
  - Use XML attribute xmlns
  - Available in EVERY xml element even if not declared in a DTD
  - xmlns is part of the XML language
  - Create namespace by including xmlns in start tag or root element of document
  - Scope of namespace is that single element and all its children
  - XHTML has a namespace
    - Designed to be easily extended using other XML DTD's
    - *Requires namespace declaration in every XHTML doc to be considered valid*

- **XHTML requires namespace**
  - Must declare in every XHTML document
    - Namespace for XHTML 1.0 is http://222.23.org/1999/xhtml
    - If no other namespaces are used or if it is the only namespace you can declare XHTML as the “default” namespace and you won’t need to prefix tags just declare it:

```
<html xmlns=“http://www.w3.org/1999/xhtml”>
...
</html>
```

    - Because there is no :name= in the xmlns declaration this becomes the “default” namespace
    - Consider which tags will be most used and make that your default or primary namespace

```
<html xmlns=“http://www.w3.org/1999/xhtml”
        xmlns:htm=“http://www.w3.org/TR/REC-html140”>
<htm:p>Some text in the HTML 4.0 namespace</htm:p>
<p>Some text in the XHTML namespace</p>
...
</html>
```

- **Namespace "Issues"**
  - Seriously deficient for internationalization
    - URI syntax depends on US-ASCII
      - If alphabet not based upon Roman alphabet then what?
      - Some accommodations already used:
        - ue for German ü as an example
      - Many languages cannot be handled
        - Cyrillic, Arabic, Japanese, Chinese, Hebrew, etc....
      - Proposals are out to extend URI to include support for UTF-8 ISO-10646 character sets
      - Problem is, now we look like strange gobldygook language:
        - http://www.ألي&#1587:.om/
      - Overriding names using namespace prefix means you get to add prefix names to your style sheet in order to use it
  - Namespaces are considered a temporary solution waiting on XML Schemas

# XML Schemas

- **Schemas are the „Next“ generation**
  - Plan is to replace DTD
  - Allow boundary and error checking
    - Structured records can be checked for being well-formed and for being valid
  - Address issue whereby XML allows information of ANY length to be in a CDATA field, database might accept only 20
  - Want to be able to specify length and nature of information
    - Eliminate having to traffic over the internet to find these kinds of “mistakes”
    - Base for starting to find info about SCHEMA would be:

http://www.w3.org/TR/NOTE-xml-schema-req

- **Schema - model for the content**
  - **Also allows constraints**
    - Where element may be used
    - When element may be used
    - Datatype constraints
      - Data size
      - Data content
  - Consider an XML DTD with zip-code

```
<!ELEMENT zip-code #PCDATA>
```

    - sets up for interesting data for the zip-code

```
<zip-code> JMW-198734588888888-xyz</zip-code >
```

    - The content is well-formed and “valid” - it is Parsed Character Data - but it is certainly not a valid ZIP CODE
    - Datatype constraints in schemas can limit content to 5-digit number or 9-digit number broken between 5 and 6 by a hyphen
      - Certainly would improve data reliability with less overhead!

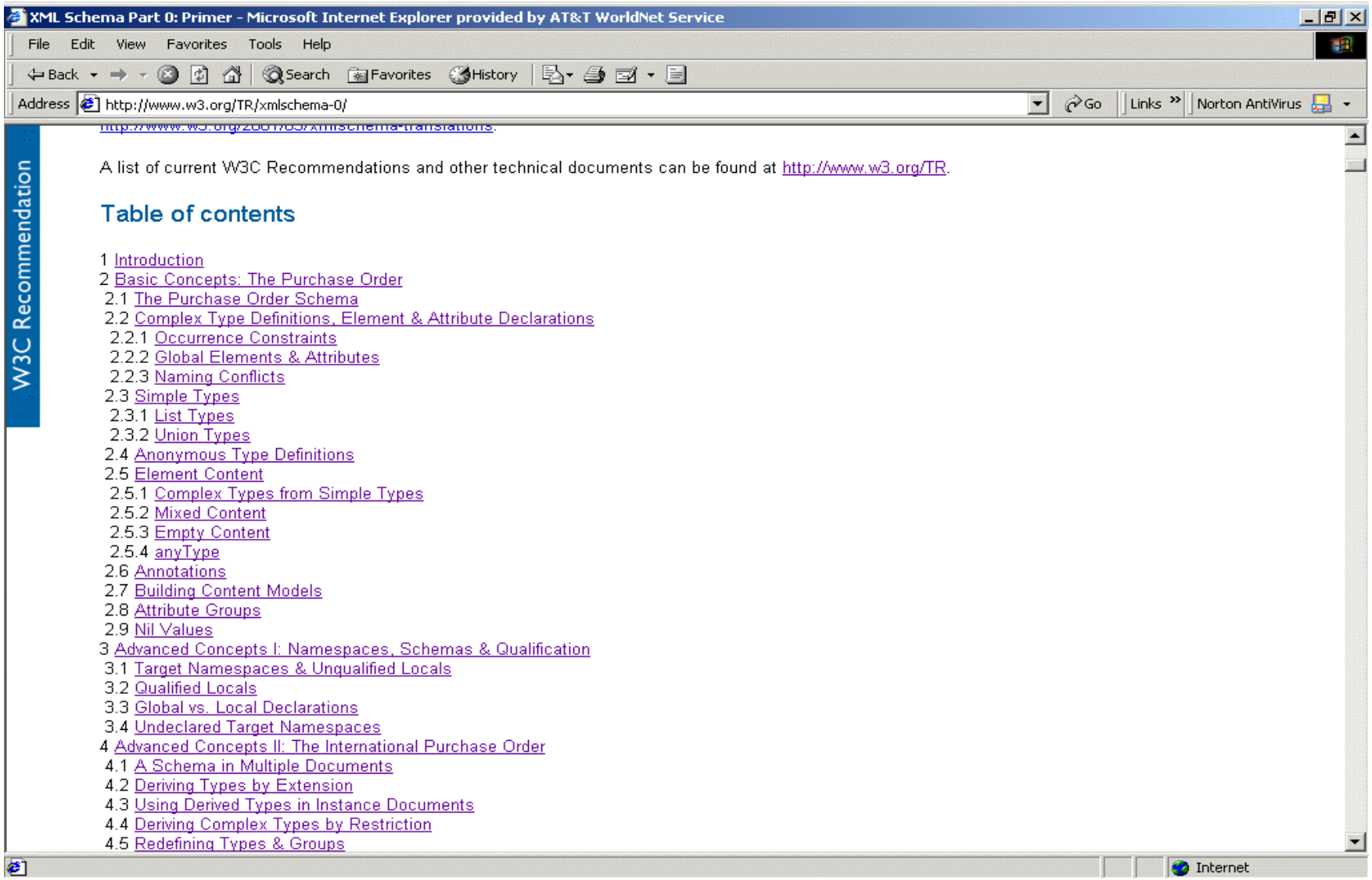
  

- **DTD and SCHEMA compared:**

  - Language used
    - DTD - EBNF
    - Schema - XML itself
  - Data constraint options
    - DTD - minimal - CDATA but not much more
      - Attributes can help with constr5aints using enumerated values
      - Not practical where large number of value choices
    - Schema - Specific constraints can be specified
  - User-Defined Types
    - DTD - limited to fixed set of content model
    - Schema - allows for flexibility in expressing content as well as in limiting content

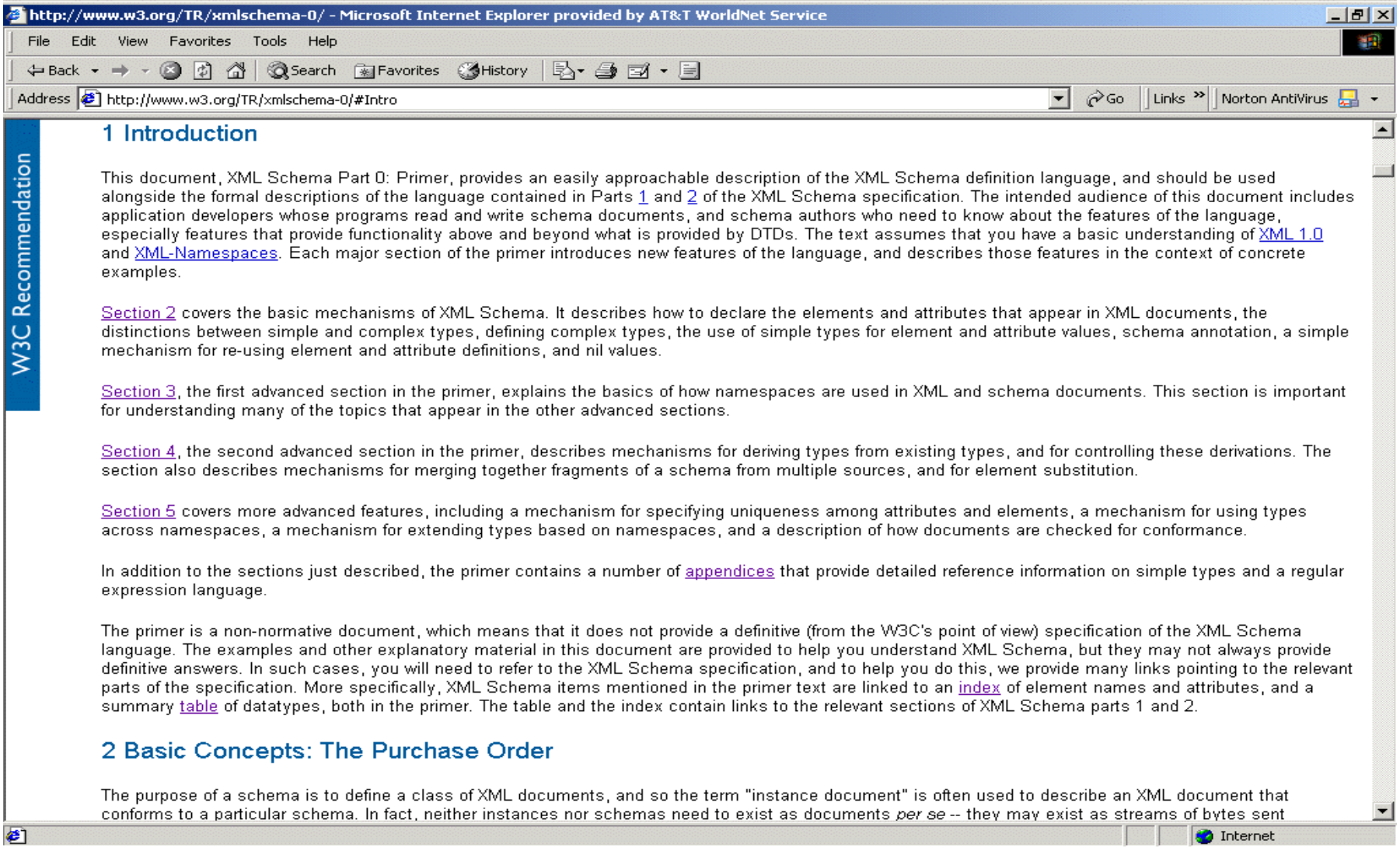
  

- **Information from:** http://www.w3.org/TR/xmlschema-0/



- **Information from:** http://www.w3.org/TR/xmlschema-0/#Intro/



- **Working group on SCHEMA describes requirements:**

1. Provide for primitive datatyping, including byte, date, integer, sequence, SQL and Java primitive datatypes, etc
2. Define type system that is adequate for import/export from database systems
3. Distinguish requirements of lexical data representation vs. requirements controlling underlying information set
4. Allow creation of user-defined datatypes
    1. May be derived from existing and which may constrain properties (range, precision, length, mask, etc)

- **Primitive datatypes built in to XML schema (Base of these 14 datatypes)**

| | |
|---|---|
| ID | NOTATION |
| IDREF | string |
| IDREFS | boolean |
| ENTITY | number |
| ENTITIES | dateTime |
| NMTOKEN | binary |
| NMTOKENS | uri |

- **Plus these added types derived from the original 14**

| | |
|---|---|
| integer | decimal |
| real | time |
| timePeriod | |

- **User-Generated datatypes**
  - Takes an existing datatype
    - Adds constraints
      - Number is a base type
        - A number which must be positive imposes a constraint
      - Decimal is a form of integer
        - A number which must be between 0.0 and 6.0 constrains value to a minimum and maximum value
    - Precision and scale can impose constraint
    - Max value allowed constraints...
      - well....you get the idea

- **XML Schema is still a working draft**
  - Use schemas with care, as standards may change
  - Schemas will “probably” replace DTD
    - More flexibility
    - More capability
  - Here is a sample where the “phone” datatype allows either 10 (with area code) or 7 (no area code) digits

```
<datatype name=“phone”>
  <basetype name=“string”/>
    <lexicalRepresentation>
        <lexical>999-999-9999</lexical>
        <lexical>999-9999</lexical>
    </lexicalRepresentation>
</datatype>
```

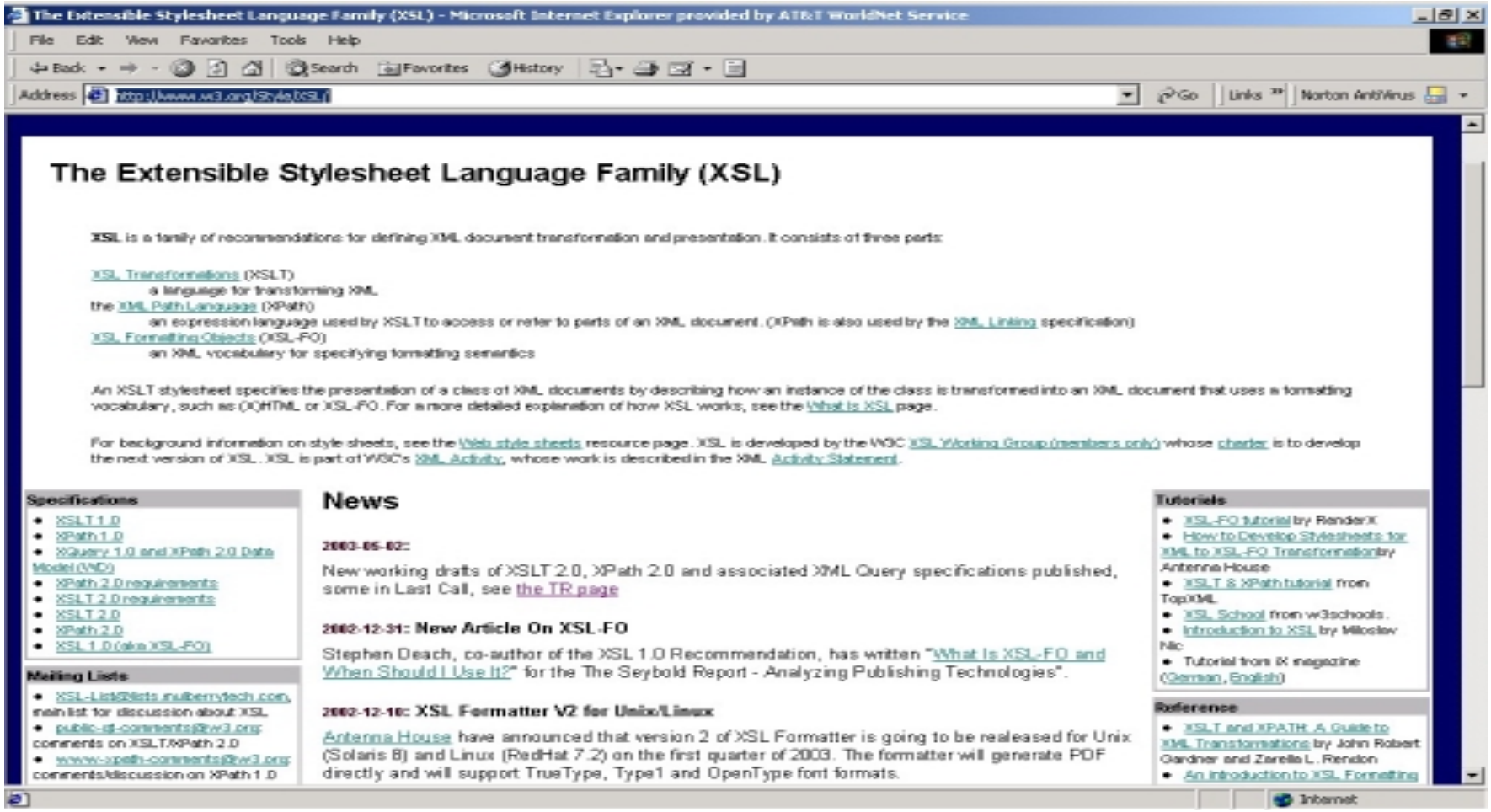
XSL

eXtensible Style sheet Language

- **Current information on XSL look to:**

http://www.w3.org/Style/XSL/

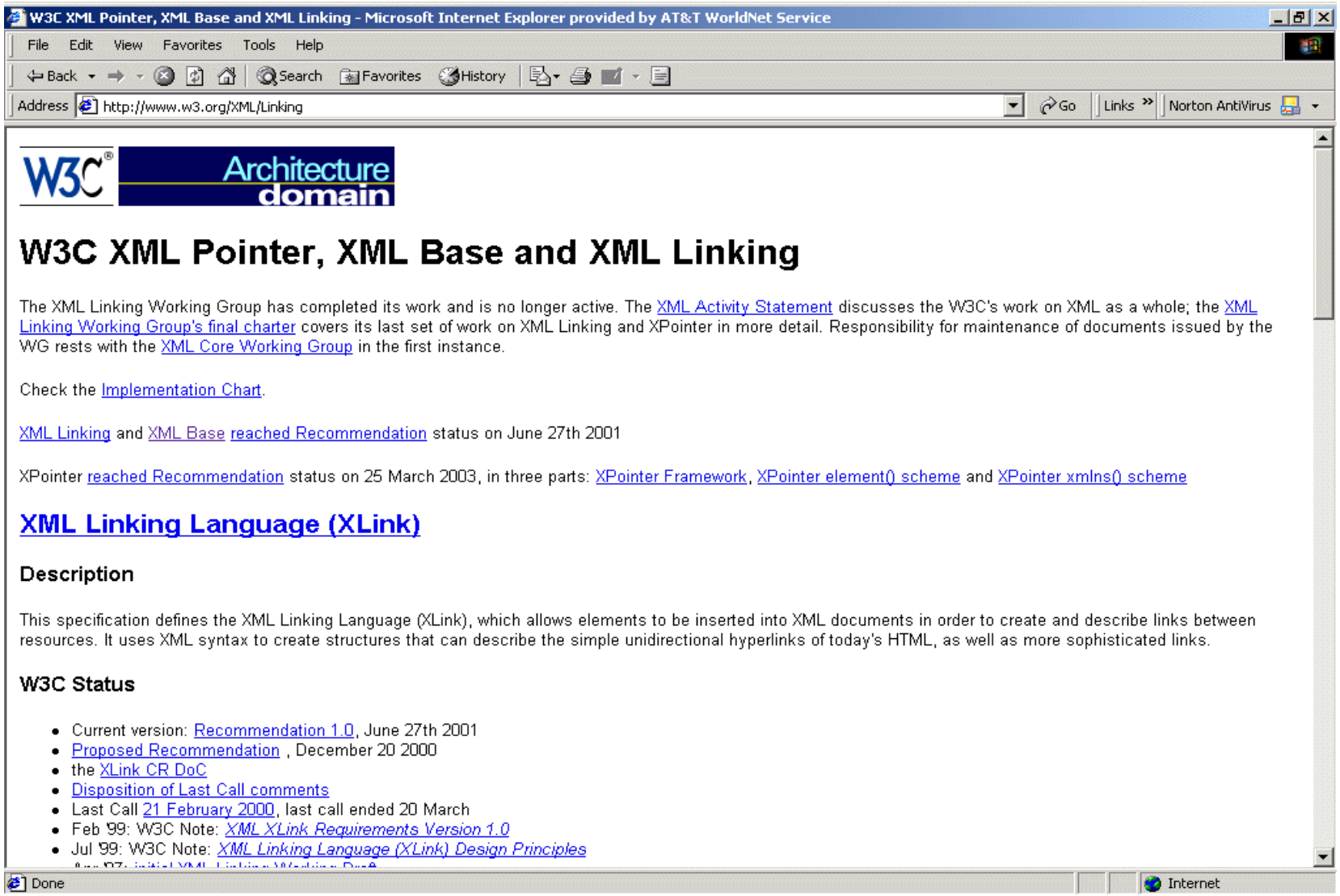


- **XSL**
  - Really 3 separate languages in various development stages
    - XML vocabulary can describe the formatting objects portion of XSL
    - XSLT or XSL transformations
      - markup vocabulary describing the TRANSFORMATION portion of XSL
    - XML Path language (XPath)
      - Language (not technically XML) used to address document fragments (portions of XML document)
      - This language also used by XSLT (also XPointer and XLink)
  - Transformations can be applied to document fragments
    - XSL searches for *pattern* and series of *one or more templates* that match the pattern
      - Returns a *result tree*
      - Allows for working with a small portion of an XML document

# XML Pointer and Linking



Additional resources

where can you go from here?

- **Obviously a LOT of information is available and much has been listed throughout this presentation**
- **Most of the information is available on the internet**

http://www.w3.org/
This would be “home”

http://www.alphaworks.ibm.com/xml/
alphaworks xml location

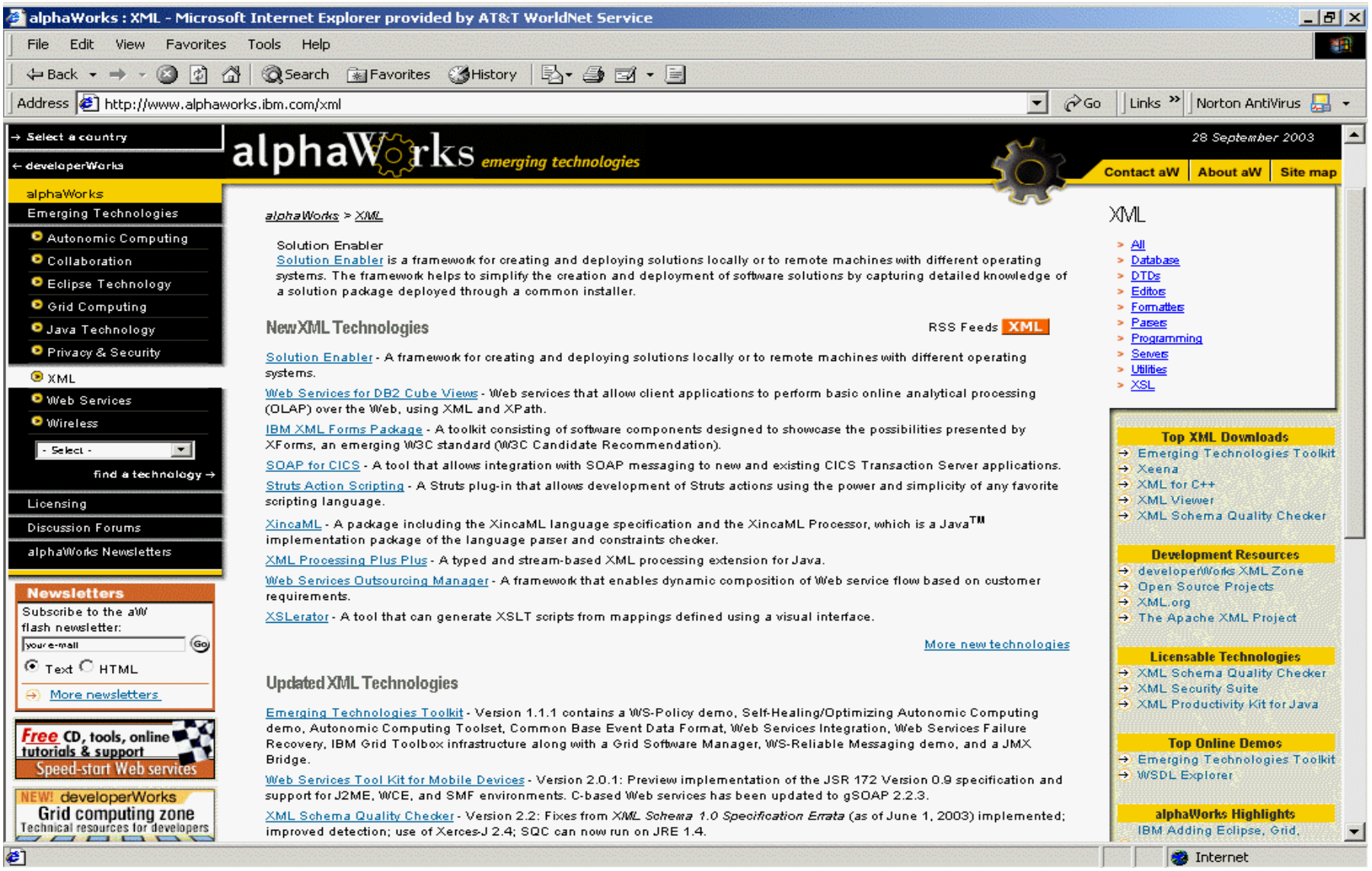
http://www-106.ibm.com/developerworks/xml/
developerworks xml site

http://www.w3schools.com/
free tutorials available for most all of the XML technologies and pieces

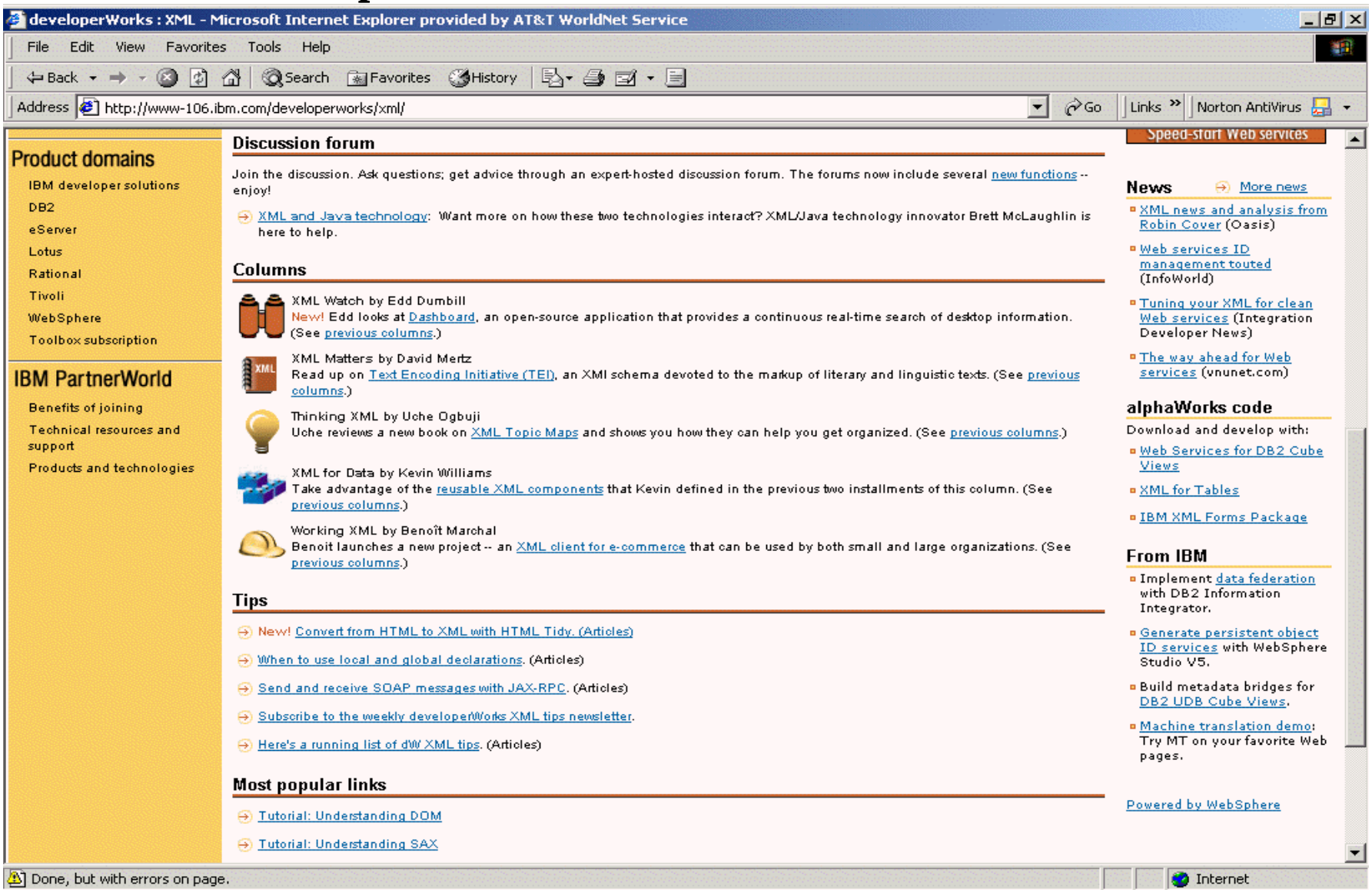
http://www.xml.com
has more tutorials as well as live web-casts

- **Check out alphaWorks.ibm.com**



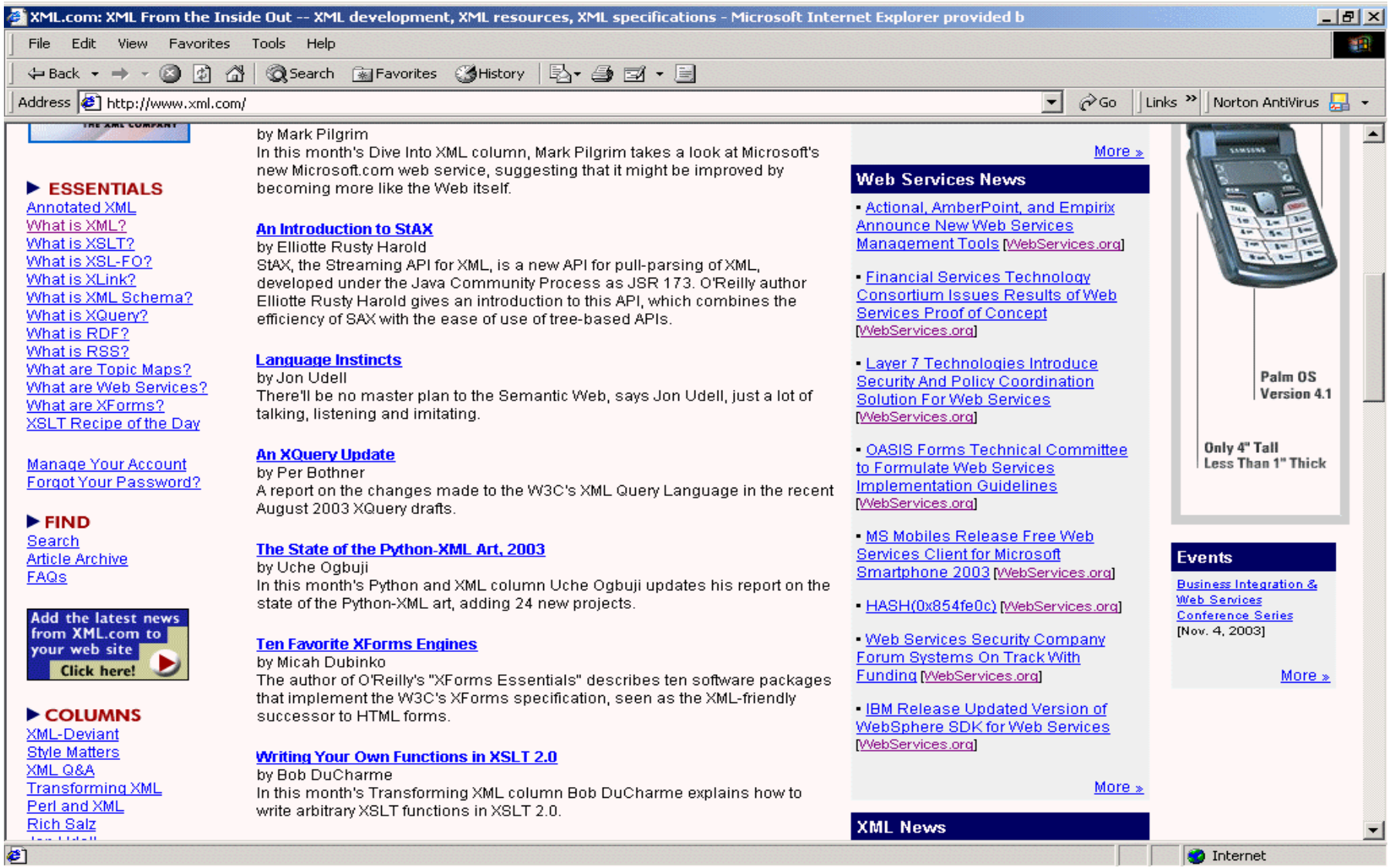
- **IBM developerWorks and XML**



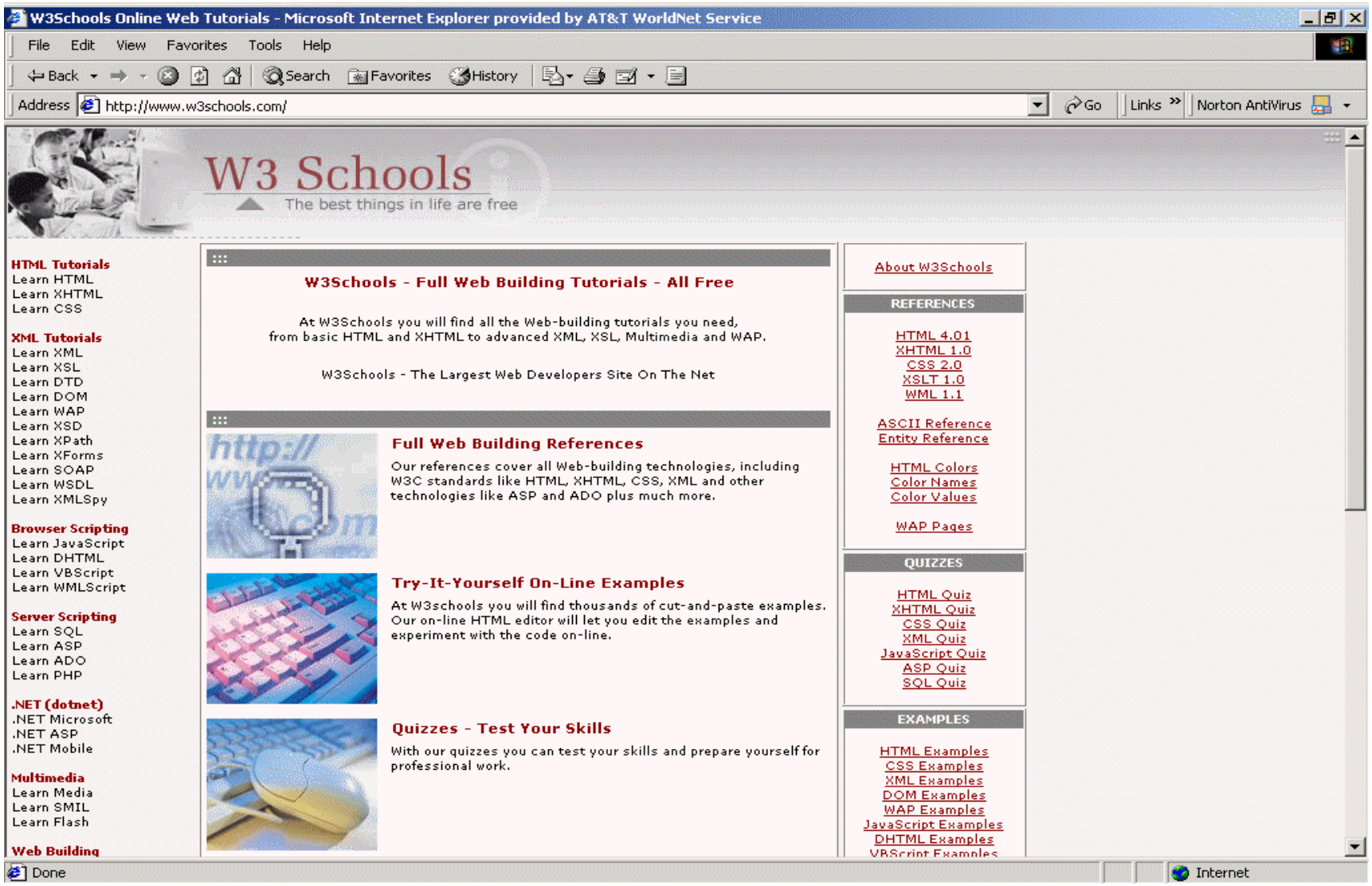

  

- **XML.COM is another good location**

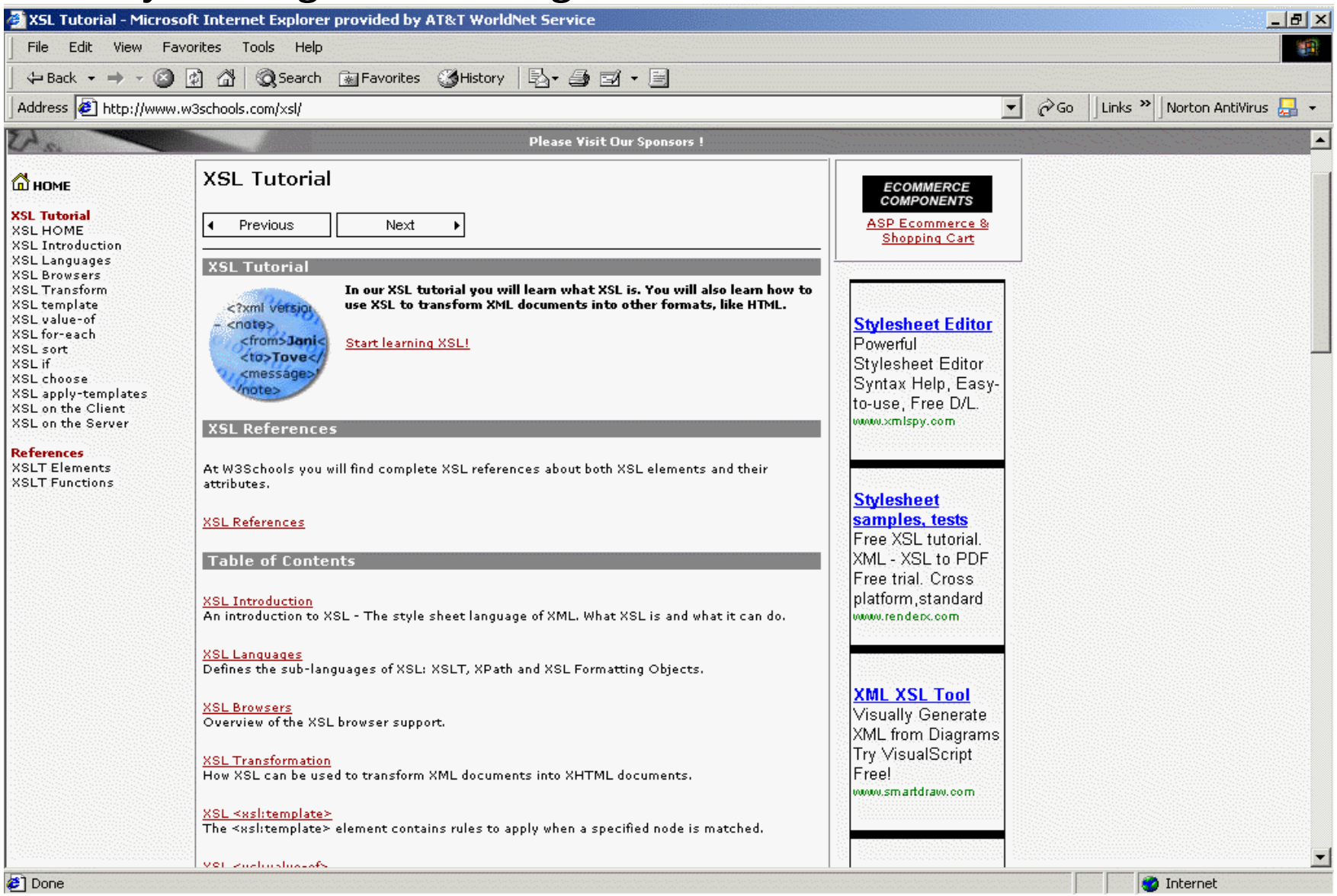
- **XML.COM is another good location**



- **This site has excellent starting tutorials!**



- **Try these good starting tutorials!**



- There is so much going on with the XML world you just have to keep checking out the www.w3.org site to see what the latest is
  - But.....you should have at least a working starting knowledge and be able to look at XML documents and understand them as well as DTD, SCHEMA, etc....etc....
    - ***XML is really just taking off***
      - ***SO KEEP YOUR SEAT BELTS FASTENED!***

RETURN TO INDEX