



XML Roadshow

Toowoomba, Mackay, Townsville, Cairns; June 2000


Bob Brown, *bob@asert.com.au*

presented by

Software Engineering Australia

in conjunction with

ASERT Consulting Pty. Ltd.



“...despite all the hype, XML is really just a new format for data stored in text files. However, its simplicity, combined with its platform and application independence, means that it is being used in an increasing number of areas where the exchange of data is required—especially between disparate systems.”



Overview

XML in Seven Points

- ★ eXtensible Markup Language is:
 - ★ a method of structuring data to facilitate open interchange formats
 - ★ like HTML but isn't HTML
 - *"HTML without the training wheels..."*
 - permits developers to "get up to speed" quickly
 - ★ text, but isn't meant to be read by humans
 - it is mostly intended for software to parse and manipulate
 - ★ actually a family of technologies
 - XSL, XLink/XPointer, DOM, Schemas, etc....
 - ★ verbose, but not inefficient
 - documents perhaps larger to store but may be more efficient to process than other formats
 - ★ new, but not that new
 - traces its roots to the 'sixties'
 - ★ licence-free, platform and vendor neutral
 - as all good technologies should be ☺

Markup Languages

- ☀ describe a document

- ☀ typically this has simply meant “make the data ‘pretty.’”

- difficult/impossible to integrate with other business processes

- ☀ XML introduces the idea of markup for processing purposes

- ☀ origins:

- ☀ ‘prehistory’: RUNOFF, [ntg]roff, etc.

- ☀ ‘70: IBM DCF/GML

- ☀ ‘86: SGML/HyTime

- ☀ others: rtf, PostScript, proprietary formats

- ☀ ~ ‘94: HTML

Goals of XML

- ★ to be easily usable over the Internet
 - or any situation requiring data interchange; the Internet is simply the most ‘trendy’ example...
 - must be simpler than SGML
- ★ to support a wide variety of applications
 - authoring tools, search engines, databases, publishing engines, etc.
- ★ to be compatible with SGML
 - allows easy adoption; compatible with government requirements; existing SGML tools can process XML easily
- ★ it must be easy to process XML
 - writing applications is easy; designers originally had the idea of a “two week” benchmark

More Goals

- ✦ should have very few optional features
 - ✦ ideally zero...
- ✦ to be human-readable
 - ✦ text based
 - makes life a lot easier for a developer
 - *may* be **marginally** less efficient
 - ✦ SGML allowed strange abbreviations and shortcuts; terseness is of minimal importance for XML
- ✦ XML's design shall be formal and concise and also prepared as quickly as possible
 - ✦ formality should make life easier for everybody
 - ✦ needs to be defined according to "Internet speed"
- ✦ XML documents shall be easy to create
 - ✦ it should also be easy to make good tools

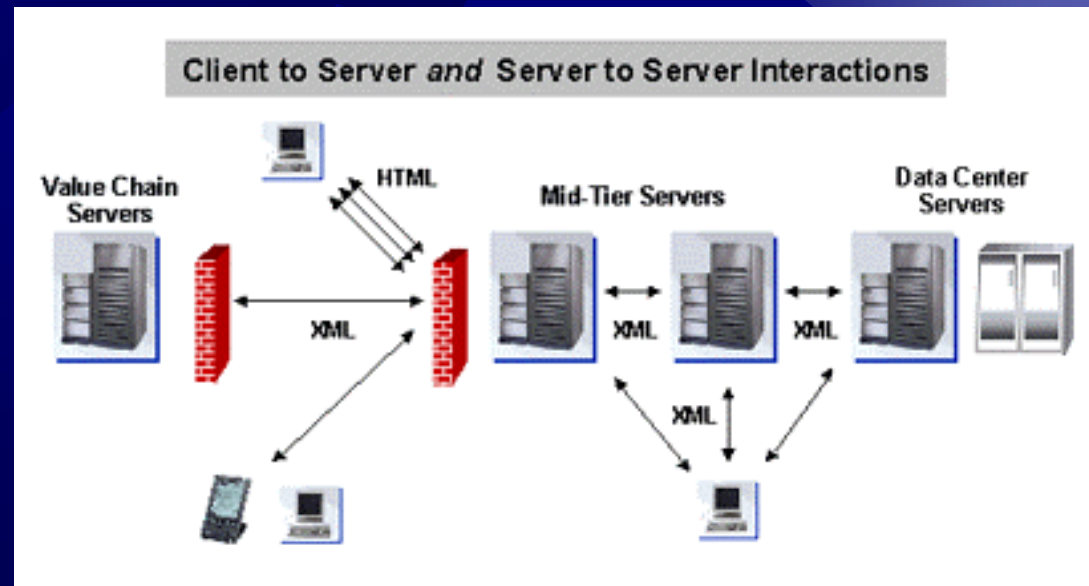
XML Standards Arena

- ★ XML development carried out under auspices of W3C
 - ★ an industry consortium, not a formal standards body
 - ★ W3C recommendations may become true standards eventually
 - *“The process...for creating a Recommendation is an alternative to, and not a replacement for,...the standards process...”*
- ★ XML is in a “state of flux”
 - ★ lots of input from many sources
 - a few false starts taken
 - ★ conflicting vendor ‘interests’
 - ★ implementing, testing and evaluating ideas takes time and effort

XML 1.0	W3C Recommendation
Namespaces	”
XSLT	”
RDF	”
XPath	”
DOM Level 1	”
DOM Level 2	W3C Candidate Recommendation
XPointer	”
Infoset	W3C Working Draft
XLink	”
XML Schema 1 & 2	”
XDR	”
SAX	Version 2.0

XML and the Enterprise

- ✦ helping to facilitate the move to “*a global information sharing society.*”
- ✦ relevant throughout a tiered enterprise architecture
 - ✦ XML offers a robust solution as the underlying architecture for data in n -tier architectures
 - ✦ DBs; B2B; U2B; workflows; documents; devices; web; etc., etc.



An Integration Technology

- ✦ excels where data has to be interchanged across **heterogeneous** boundaries
- ✦ arguably of minimal benefit for homogeneous systems
 - ✦ why reinvent the wheel?
- ✦ doesn't assume strong coupling
 - ✦ unlike EDI
 - everything is achieved through interchange of documents
 - flexible enough for open market "e-commerce"
- ✦ *"The ASCII of the Future"*
 - ✦ according to Microsoft, that is...

The background is a dark blue field filled with various sizes of gear silhouettes. On the left side, there is a vertical strip containing a detailed, colorful image of interlocking gears in shades of orange, yellow, and brown.

Basic XML Syntax

```

<?xml version="1.0" ?>
<!DOCTYPE main [
<!ELEMENT main (purchase)*>
<!ELEMENT purchase (date, account, item+)>
<!ELEMENT date (#PCDATA)>
<!ELEMENT account (#PCDATA)>
<!ELEMENT item (itemno, itemdes, quantity)>
<!ELEMENT itemno (#PCDATA)>
<!ELEMENT itemdes (#PCDATA)>
<!ELEMENT quantity (#PCDATA)>
]>
<main>
  <purchase>
    <date>19-September-1999</date>
    <account>Fred_Flintstone</account>
    <item>
      <itemno>478B</itemno>
      <itemdes>3 1/2 Floppy Disk</itemdes>
      <quantity>1000</quantity>
    </item>
    <item>
      <itemno>6937A</itemno>
      <itemdes>Mouse Pad</itemdes>
      <quantity>50</quantity>
    </item>
  </purchase>
</main>

```

The screenshot shows a Microsoft Internet Explorer window titled "C:\TEMP\x.xml - Microsoft Internet Explorer". The address bar contains "C:\TEMP\x.xml". The main content area displays the XML document with the following structure:

```

<?xml version="1.0" ?>
<!DOCTYPE purchases (View Source for full doctype...)>
- <main>
- <purchase>
  <date>19-September-1999</date>
  <account>Fred_Flintstone</account>
- <item>
  <itemno>478B</itemno>
  <itemdes>3 1/2 Floppy Disk</itemdes>
  <quantity>1000</quantity>
</item>
- <item>
  <itemno>6937A</itemno>
  <itemdes>Mouse Pad</itemdes>
  <quantity>50</quantity>
</item>
</purchase>
</main>

```

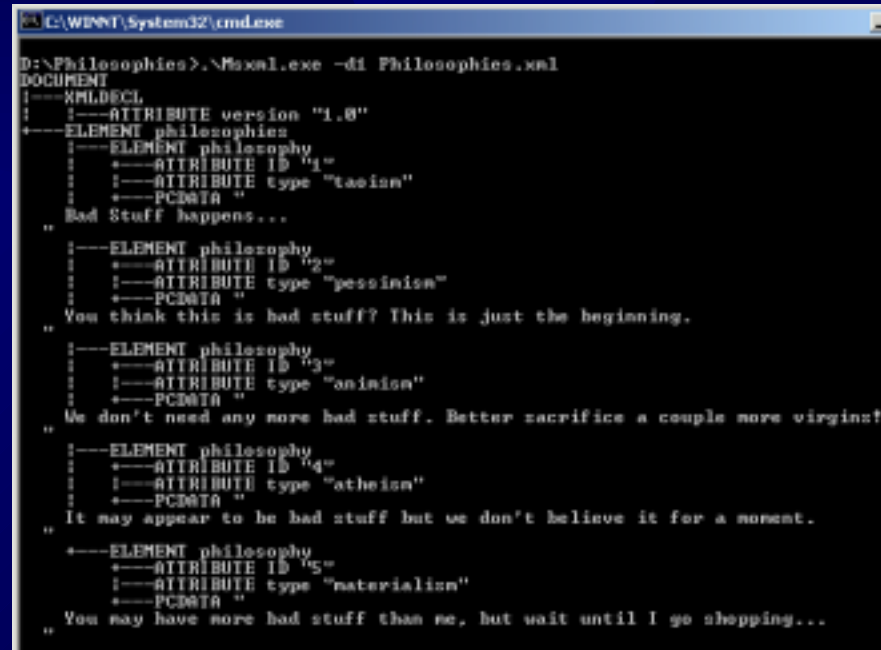
The status bar at the bottom shows "Done" and "My Computer".

Document Tree...

✦ a document is a tree

✦ perfect nesting required

```
<?xml version="1.0"?>
<philosophies>
  <philosophy ID="1" type="taoism">
    Bad Stuff happens...
  </philosophy>
  <philosophy ID="2" type="pessimism">
    You think this is bad stuff? This is just the beginning.
  </philosophy>
  <philosophy ID="3" type="animism">
    We don't need any more bad stuff. Better sacrifice a couple more virgins!
  </philosophy>
  <philosophy ID="4" type="atheism">
    It may appear to be bad stuff but we don't believe it for a moment.
  </philosophy>
  <philosophy ID="5" type="materialism">
    You may have more bad stuff than me, but wait until I go shopping...
  </philosophy>
</philosophies>
```



```
C:\WINNT\System32\cmd.exe
D:\Philosophies>.\\Msxml.exe -di Philosophies.xml
DOCUMENT
|--XMLDECL
|  |--ATTRIBUTE version "1.0"
|--ELEMENT philosophies
|  |--ELEMENT philosophy
|  |  |--ATTRIBUTE ID "1"
|  |  |--ATTRIBUTE type "taoism"
|  |  |--PCDATA "
|  |  |Bad Stuff happens...
|  |  "
|  |--ELEMENT philosophy
|  |  |--ATTRIBUTE ID "2"
|  |  |--ATTRIBUTE type "pessimism"
|  |  |--PCDATA "
|  |  |You think this is bad stuff? This is just the beginning.
|  |  "
|  |--ELEMENT philosophy
|  |  |--ATTRIBUTE ID "3"
|  |  |--ATTRIBUTE type "animism"
|  |  |--PCDATA "
|  |  |We don't need any more bad stuff. Better sacrifice a couple more virgins!
|  |  "
|  |--ELEMENT philosophy
|  |  |--ATTRIBUTE ID "4"
|  |  |--ATTRIBUTE type "atheism"
|  |  |--PCDATA "
|  |  |It may appear to be bad stuff but we don't believe it for a moment.
|  |  "
|  |--ELEMENT philosophy
|  |  |--ATTRIBUTE ID "5"
|  |  |--ATTRIBUTE type "materialism"
|  |  |--PCDATA "
|  |  |You may have more bad stuff than me, but wait until I go shopping...
|  |  "
|--PCDATA ""
```

XML Structure

- ★ XML document composed of:

- ★ prologue, elements, entities, processing instructions and comments
 - some are optional, some required
 - an important aspect is the possibility of creating *self-describing* documents
- ★ may also contain processing code

```
<?xml version="1.0" ?>
<HAIKU xml:space="preserve">
I'm sorry, there's -- um --
    insufficient -- what's-it-called?
        The term eludes me ...
-- Owen Mathews
</HAIKU>
```


Prologue, Element & Attribute

☀ prologue

```
<?xml version="1.0" encoding="UTF-8">
```

- ☀ tells the xml processor about the data

☀ element

- ☀ container for data

☀ attribute

- ☀ associated data or property of container

Attribute

┌──────────┐

Container Elements { <SHOE STOCK_ID="X19E435">

{ <COLOUR>

BROWN

{ </COLOUR>

{ <SIZE>

43 Wide

{ </SIZE>

{ </EVENT>

Elements & Attributes are *more-or-less* interchangeable

Processing Instruction

- ☀ commands or information passed straight to the application that is processing the XML data
 - ☀ ignored by XML itself

```
<PARA>  
  this element also contains two processing instructions (to allow for  
  two different processing applications)  
  <?javascript do something for javascript ?>  
  <?perlscript do something equivalent for perlscript ?>  
</PARA>
```

- ☀ the target name “xml” is reserved for use by XML itself

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

Comment; CDATA; Entity

- ☀ comments facilitate human comprehension

```
<!-- this is a comment -->
```

- ☀ CDATA for 'awkward' data

- ☀ entities mostly provide a simple 'macro' facility

```
<?xml version="1.0"?>  
<PROGRAM lang="BASIC">  
10 LET A=10  
20 LET B=20  
30 IF A <![CDATA[<]]>B THEN PRINT A+B  
</PROGRAM>
```

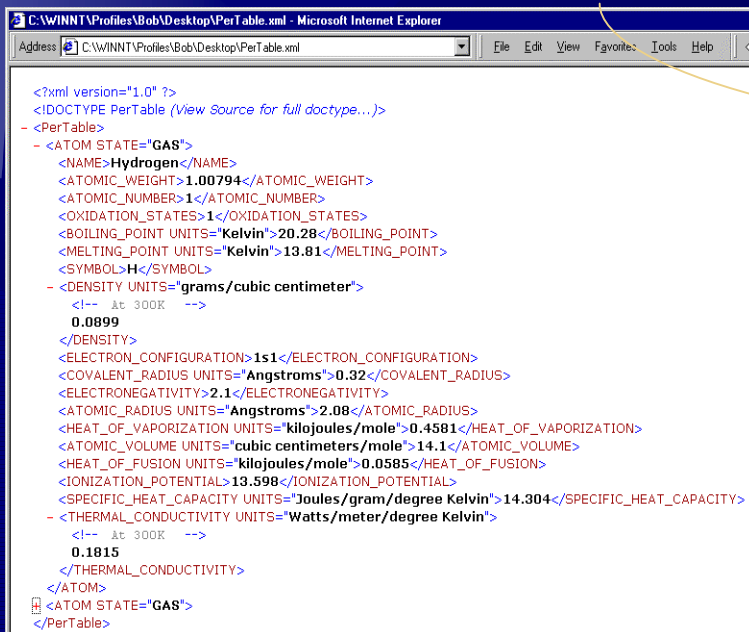
```
<?xml version="1.0" ?>  
<!DOCTYPE MAIN [  
<!ENTITY bob "Bob Brown">  
]>  
<MAIN>  
  &lt; &#169; &gt; &bob;, Transentia Pty. Ltd., 2000  
</MAIN>
```



External Entity

☀ provides a means to incorporate external data

```
<ATOM STATE="GAS">
  <NAME>Hydrogen</NAME>
  <ATOMIC_WEIGHT>1.00794</ATOMIC_WEIGHT>
  <ATOMIC_NUMBER>1</ATOMIC_NUMBER>
  <OXIDATION_STATES>1</OXIDATION_STATES>
  ...
</ATOM>
```



The screenshot shows a web browser window displaying an XML document. The address bar shows the file path: C:\WINNT\Profiles\Bob\Desktop\PerTable.xml. The XML content includes a DOCTYPE declaration for 'PerTable' and a series of external entity references for various elements like 'Hydrogen', 'Helium', etc. The document structure is as follows:

```
<?xml version="1.0" ?>
<!DOCTYPE PerTable (View Source for full doctype...)
- <PerTable>
- <ATOM STATE="GAS">
  <NAME>Hydrogen</NAME>
  <ATOMIC_WEIGHT>1.00794</ATOMIC_WEIGHT>
  <ATOMIC_NUMBER>1</ATOMIC_NUMBER>
  <OXIDATION_STATES>1</OXIDATION_STATES>
  <BOILING_POINT UNITS="Kelvin">20.28</BOILING_POINT>
  <MELTING_POINT UNITS="Kelvin">13.81</MELTING_POINT>
  <SYMBOL>H</SYMBOL>
- <DENSITY UNITS="grams/cubic centimeter">
  <!-- At 300K -->
  0.0899
</DENSITY>
<ELECTRON_CONFIGURATION>1s1</ELECTRON_CONFIGURATION>
<COVALENT_RADIUS UNITS="Angstroms">0.32</COVALENT_RADIUS>
<ELECTRONEGATIVITY>2.1</ELECTRONEGATIVITY>
<ATOMIC_RADIUS UNITS="Angstroms">2.08</ATOMIC_RADIUS>
<HEAT_OF_VAPORIZATION UNITS="kilojoules/mole">0.4581</HEAT_OF_VAPORIZATION>
<ATOMIC_VOLUME UNITS="cubic centimeters/mole">14.1</ATOMIC_VOLUME>
<HEAT_OF_FUSION UNITS="kilojoules/mole">0.0585</HEAT_OF_FUSION>
<IONIZATION_POTENTIAL>13.598</IONIZATION_POTENTIAL>
<SPECIFIC_HEAT_CAPACITY UNITS="Joules/gram/degree Kelvin">14.304</SPECIFIC_HEAT_CAPACITY>
- <THERMAL_CONDUCTIVITY UNITS="Watts/meter/degree Kelvin">
  <!-- At 300K -->
  0.1815
</THERMAL_CONDUCTIVITY>
</ATOM>
<ATOM STATE="GAS">
</PerTable>
```

```
<ATOM STATE='GAS'>
  <NAME>Helium</NAME>
  <ATOMIC_WEIGHT>4.0026</ATOMIC_WEIGHT>
  <ATOMIC_NUMBER>2</ATOMIC_NUMBER>
  <BOILING_POINT UNITS="Kelvin">4.216</BOILING_POINT>
  <MELTING_POINT UNITS="Kelvin">0.95</MELTING_POINT>
  <SYMBOL>He</SYMBOL>
  <DENSITY UNITS="grams/cubic centimeter">
    <!-- At 300K -->
    0.1785
  </DENSITY>
  <ELECTRON_CONFIGURATION>1s2</ELECTRON_CONFIGURATION>
  ...
  <THERMAL_CONDUCTIVITY
    UNITS="Watts/meter/degree Kelvin">
    <!-- At 300K -->
    0.152
  </THERMAL_CONDUCTIVITY>
</ATOM>
```

```
<!-- file: Elements.xml -->
<!ENTITY H SYSTEM "h.xml">
<!ENTITY He SYSTEM "He.xml">
```

```
<?xml version="1.0" ?>
<!DOCTYPE PerTable [
  <!ENTITY % Elements SYSTEM "Elements.xml">
  %Elements;
]>
<PerTable>
  &H;
  &He;
</PerTable>
```

The background is a dark blue field filled with various sizes of gear silhouettes. On the left side, there is a vertical strip containing a detailed, colorful image of interlocking gears in shades of orange, yellow, and brown. The main title 'Data Modelling' is centered in a large, white, sans-serif font.

Data Modelling

The DTD

- ✱ XML's data modelling facility
- ✱ specified by the Document Type Declaration at the top of a file
- ✱ specifies the logical structure of a document
 - ✱ *structure*, not semantics or constraints on content data
 - ✱ allows a processing application to determine whether the data contained in an XML is *structured* correctly:
 - no missing (required) data/attributes
 - no extra (unexpected) data
 - relationships between different parts is correct
- ✱ creating “self describing documents”

Why Have a DTD?

★ well-formed versus valid XML

- ★ a **well-formed** document is structurally sound but may contain other errors
 - missing elements, attributes, entities; an unbalanced document tree; duplicate IDs that should be unique, etc.
 - an XML document without an accompanying DTD can only be checked for “well formed-ness”
- ★ DTD allows correspondence between physical and logical structure to be checked
- ★ a **valid** document must be well-formed, plus the contents of the document must conform to the rules specified in the DTD
 - only if there is an associated DTD can conformance be checked: “...unlike HTML, the built-in validity checking of XML allows users to trust the data. Validity checking makes XML appropriate for transactions, electronic commerce and inventory management.”

DTD Example

Notables:

- DOCTYPE / root element correspondence
- structure symbols
- simple data types
- special notation for empty elements

```
<!ELEMENT EMAIL (TO+, FROM, CC*,  
                BCC*, SUBJECT?, BODY?)>
```

```
<?xml version="1.0"?>  
<!DOCTYPE FAMILYTREE [  
<!ELEMENT FAMILYTREE (PERSON*)>  
<!ELEMENT PERSON (NAME, SPOUSE*)>  
<!ATTLIST PERSON  
              NUM ID #REQUIRED  
              FATHER IDREF #IMPLIED  
              MOTHER IDREF #REQUIRED  
>  
<!ELEMENT NAME (#PCDATA)>  
<!ELEMENT SPOUSE EMPTY>  
<!ATTLIST SPOUSE  
              NUM IDREFS #IMPLIED>  
>  
<FAMILYTREE>  
  <PERSON NUM="p1" FATHER="p3" MOTHER="p4">  
    <NAME>Bob Brown</NAME>  
    <SPOUSE NUM="p2"/>  
  </PERSON>  
  <PERSON NUM="p2" FATHER="p5" MOTHER="p6">  
    <NAME>Semmi Sin</NAME>  
    <SPOUSE NUM="p1"/>  
  </PERSON>  
  <PERSON NUM="p3">  
    <NAME>Charlie Brown</NAME>  
    <SPOUSE NUM="p4"/>  
  </PERSON>  
  <PERSON NUM="p4">  
    <NAME>Marion Brown</NAME>  
    <SPOUSE NUM="p3"/>  
  </PERSON>  
  ...
```


Schemas

- ✦ an XML vocabulary
- ✦ overcome the (many!) deficiencies of DTDs

“While XML 1.0 supplies a mechanism, the Document Type Definition (DTD) for declaring constraints on the use of markup, automated processing of XML documents requires more rigorous and comprehensive facilities in this area. Requirements are for constraints on how the component parts of an application fit together, the document structure, attributes, datatyping, and so on. The W3C XML Schema Working Group is addressing means for defining the structure, content and semantics of XML documents.”

- ✦ DTDs have:
 - very limited capability to describe the *content* of a document
 - only concerned with its structure
 - weak data typing: pretty much just TEXT
 - an important issue for an application such as dumping/restoring an SQL database
 - odd syntax
 - effectively a separate language
 - increases complexity of parsers

Goals

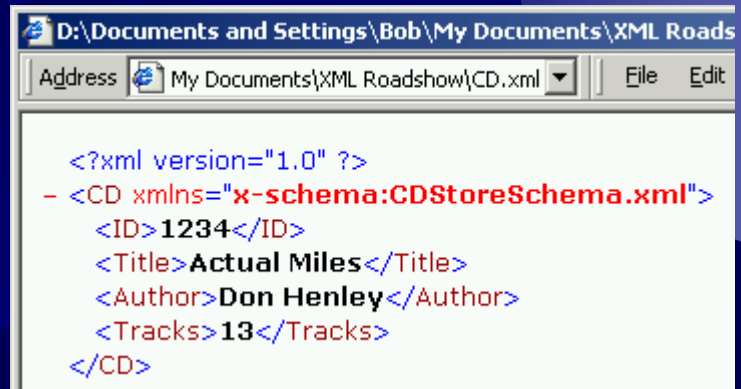
- ✦ authors shouldn't have to learn a new syntax
 - ✦ the schema language should be expressed in XML; in comparison, a normal DTD has a special syntax and so requires special treatment
 - ✦ also makes life much easier for processing tools
 - encourages uptake and allows XML to exist at "Internet speed"
- ✦ schemas should be extensible
 - ✦ allow for flexible data modelling techniques (such as OOP)
- ✦ schemas should meet the needs of applications requiring extensive data validation
 - ✦ such as database interchange; need proper data types and constraints on data values, etc.

More Goals

- ✦ a schema must be able to be built up from parts coming from many sources
 - ✦ so that a document can be constructed that incorporates several specifications and so meets many requirements
- ✦ schemas should encourage reuse
 - ✦ DTDs can allow reuse but things quickly get messy; schemas offer a simpler solution
- ✦ schemas should be upwardly compatible with XML 1.0
 - ✦ minimise “technology churn”
 - ✦ standard DTDs can still be used, if an application has simple enough requirements

```
<?xml version="1.0" ?>
<Schema name="CDStoreSchema"
  xmlns="urn:schemas-microsoft-com:xml-data"
  xmlns:dt="urn:schemas-microsoft-com:datatypes">
  <ElementType name="ID" dt:type="string" />
  <ElementType name="Title" dt:type="string" />
  <ElementType name="Author" dt:type="string" />
  <ElementType name="Tracks" dt:type="string" />
  <ElementType name="CD" model="closed">
    <element type="ID" />
    <element type="Title" />
    <element type="Author" />
    <element type="Tracks" />
  </ElementType>
</Schema>
```

```
<?xml version="1.0" ?>
<CD xmlns="x-schema:CDStoreSchema.xml">
  <ID>1234</ID>
  <Title>Actual Miles</Title>
  <Author>Don Henley</Author>
  <Tracks>13</Tracks>
</CD>
```



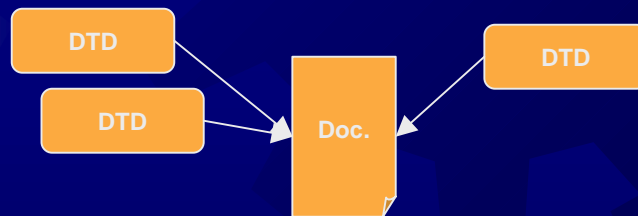
```
<?xml version="1.0" ?>
- <CD xmlns="x-schema:CDStoreSchema.xml">
  <ID>1234</ID>
  <Title>Actual Miles</Title>
  <Author>Don Henley</Author>
  <Tracks>13</Tracks>
</CD>
```



Other Features

Namespaces

- ✦ attempt to make the definition of elements/attributes globally unique
 - ✦ simplify the process of combining portions of different DTDs: lets you write an XML document that uses two or more sets of XML tags in modular fashion



- ✦ a simple idea
 - give things a (hopefully!) globally unique name
 - then `<law:bill>` and `<currency:bill>` can define different bills without conflict...
 - similar to Java's packages

Namespaces

- ☀ each element/attribute is associated with/grouped into a space which is named by a URI

```
<?xml version="1.0" ?>
<BOOKS>
  <bk:BOOK xmlns:bk="urn:BookLovers.org:BookInfo"
           xmlns:money="urn:Finance:Money">
    <bk:TITLE>A Suitable Boy</bk:TITLE>
    <bk:PRICE money:currency="US Dollar">22.95</bk:PRICE>
    <bk:COMMENT>A damn good read!</bk:COMMENT>
  </bk:BOOK>
</BOOKS>
```

- ☀ a bit confusing
- ☀ controversial

From <http://www.xml.com/pub/1999/01/namespaces.html>:

“What Do Namespace Names Point At?”

One of the confusing things about all this is that namespace names are URLs; it's easy to assume that since they're Web addresses, they must be the address of something. They're not; these are URLs, but the namespace draft doesn't care what (if anything) they point at. Think about the example of the XML.com programmer looking for book titles; that works fine without the namespace name pointing at anything.

The reason that the W3C decided to use URLs as namespace names is that they contain domain names (e.g. www.xml.com), which work globally across the Internet.”

Links, Paths and Pointers

- ★ *“A revolution in the way documents can be linked.”*
- ★ three proposals being worked on; each solves a different part of the problem
 - XML Path Language: XPath
 - a comprehensive language for document addressing
 - helps obviate the need for predefined structures and targets
 - provides a foundation for the next two...
 - XML Pointer Language: XPointer
 - extends Xpath for use in URIs
 - also introduces the ideas of points and ranges in an XML document
 - XML Linking Language: XLink
 - a vocabulary allowing the definition of suites of documents
- ★ development slowly gathering pace after a long delay
- ★ not really supported by any ‘real’ piece of software
 - IE5 supports XPath

Solving Problems

- ✦ some problems are inherent in the HTML-style linking mechanism:
 - ✦ HTML links are not self-descriptive
 - you have no idea what a link will do until it is actuated
 - ✦ HTML links waste bandwidth when only a portion of a target document is needed
 - need to retrieve whole document and then process it
 - ✦ HTML links can only return a single target resource
 - compare this with many 'help' applications that can present multiple targets
 - ✦ HTML links are too closely coupled to the structure of the target document
 - a maintenance nightmare; lots of broken links
 - difficult for groups of documents to be worked on in isolation

XPath Patterns

- find all author elements anywhere within the current document:

```
//author
```

- find all bookstores where the value of the specialty attribute is equal to “textbooks”:

```
/bookstore[@specialty = "textbooks"]
```

- find all books where the value of the style attribute on the book is equal to the value of the specialty attribute of the bookstore element at the root of the document:

```
book[/bookstore/@specialty = @style]
```

```
<xsl:if match="vehicle[@type='sports']">  
  Sports Car  
</xsl:if>
```

XPointer

- ✦ a tool for referencing portions of documents, built on the clean tree structures of XML documents
 - ✦ makes it possible to describe a path through a document tree structure
 - ✦ can specify single or multiple locations in a target document
 - ✦ vocabulary defines five *location terms* (addressing styles/mechanisms):
 - absolute; relative; string; attribute; span

```
<simpleLink
  xmlns:xlink="http://www.w3.org/2000/xlink"
  xlink:type="simple"
  xlink:href="doc.xml#xpointer(book/chapter position() &lt;= 5)" />
```

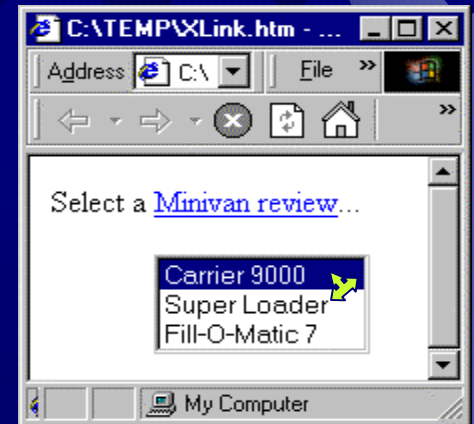
XLink

- a flexible vocabulary for connecting documents and document fragments
 - an arbitrary element can be specified as a link by applying the special *xml:link* attribute
- the traversal mechanism of an XLink is left to an associated style sheet to specify
 - *neither CSS Nor XSL currently provide facilities that can be used to define such a mechanism!*

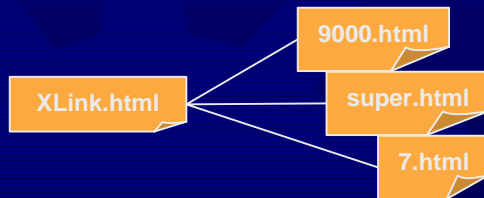
```

<link xml:link="extended" show="replace" actuate="user">
  <xl:locator href="sample.xml" role="data"/>
  <xl:locator href="sample.xsl" role="stylesheet"/>
  <xl:locator href="sample.sch" role="schema"/>
  Replace the contents of the current window with the xml data,
  using the specified stylesheet and schema...
</link>

```



(simulated)



links may be bidirectional

```

<elink xml:link="extended">
  Minivan review
  <elocator title="Carrier 9000" href="9000.html" />
  <elocator title="Super Loader" href="super.html" />
  <elocator title="Fill-O-Matic 7" href="7.html" />
</elink>

```

```
<?xml version="1.0"?>
<SLIDESHOW>
  <SLIDE TITLE="Welcome to the slide show!">
    <BUTTON xml:link="simple" href="origin().following(1,SLIDE)">
      Next
    </BUTTON>
  </SLIDE>
  <SLIDE TITLE="This is the second slide">
    <BUTTON xml:link="simple" href="origin().preceding(1,SLIDE)">
      Previous
    </BUTTON>
    <BUTTON xml:link="simple" href="origin().following(1,SLIDE)">
      Next
    </BUTTON>
  </SLIDE>
  <SLIDE TITLE="This is the second slide">
    <BUTTON xml:link="simple" href="origin().preceding(1,SLIDE)">
      Previous
    </BUTTON>
    <BUTTON xml:link="simple" href="origin().following(1,SLIDE)">
      Next
    </BUTTON>
  </SLIDE>
  ...
  <SLIDE TITLE="This is the last slide">
    <BUTTON xml:link="simple" href="origin().preceding(1,SLIDE)">
      Previous
    </BUTTON>
  </SLIDE>
</SLIDESHOW>
```

XML with Style

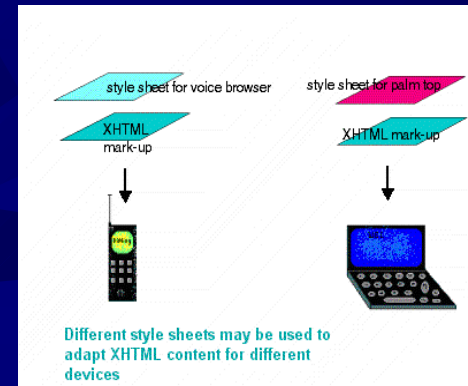
- ✦ xml:stylesheet
- ✦ two complementary additional technologies provide rendering mechanisms

- Cascading Style Sheets

- introduced for HTML
 - support is still patchy, however

- XSL

- an XML-specific technology
 - very new and in constant state of change



From <http://www.w3c.org/Style/CSS-vs-XSL.html>:

“Why does W3C recommend two different style languages?

...

Use CSS when you can, use XSL when you must.

...

CSS is much easier to use, thus easier to maintain and cheaper. ... Some things you cannot do with CSS, or with CSS alone. Then you need XSL, or at least the transformation part of XSL”

CSS & XML

☀ a no-brainer...

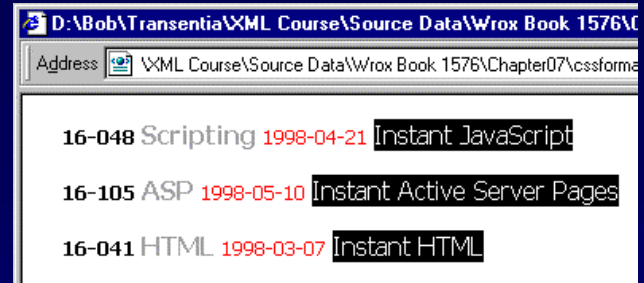
```
ITEM      { display:block; margin:15px }
CODE      { display:inline;
           font-family:Tahoma,Arial,sans-serif;
           font-size:10pt;
           font-weight:bold }

CATEGORY  { display:inline;
           font-family:Tahoma,Arial,sans-serif;
           color:darkgray;
           font-size:12pt;
           font-weight:bold }

RELEASE   { display:inline;
           font-family:Tahoma,Arial,sans-serif;
           color:red;
           font-size:10pt }

TITLE     { display:inline;
           font-family:Tahoma,Arial,sans-serif;
           font-size:12pt;
           color:white;
           background-color:black }

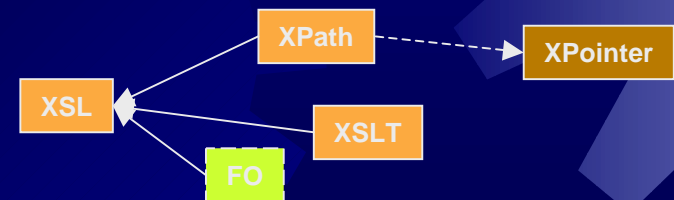
SALES     { display:none }
```



```
<?xml version="1.0"?>
<?xml:stylesheet
      type="text/css"
      href="booklist.css"?>
<BOOKLIST>
  <ITEM>
    <CODE>16-048</CODE>
    <CATEGORY>Scripting</CATEGORY>
    <RELEASE>1998-04-21</RELEASE>
    <TITLE>Instant JavaScript</TITLE>
    <SALES>375298</SALES>
  </ITEM>
  <ITEM>
    <CODE>16-105</CODE>
    <CATEGORY>ASP</CATEGORY>
    <RELEASE_>1998-05-10</RELEASE>
    <TITLE>Instant ASP</TITLE>
    <SALES>297311</SALES>
  </ITEM>
  <ITEM>
    <CODE>16-041</CODE>
    <CATEGORY>HTML</CATEGORY>
    <RELEASE>1998-03-07</RELEASE>
    <TITLE>Instant HTML</TITLE>
    <SALES>127853</SALES>
  </ITEM>
</BOOKLIST>
```


XML & XSL

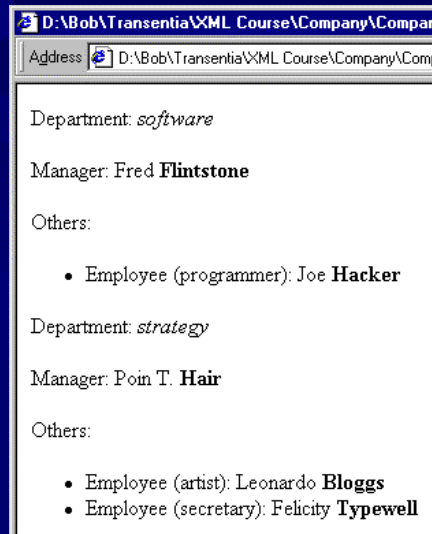
- ✦ a vocabulary for expressing stylesheets
- ✦ various components and relationships
 - ✦ XSL Transformations (XSLT)
 - transforms one XML document tree into another XML document tree
 - specifies patterns and applies templates to the matches
 - ✦ Formatting objects
 - concerned with rendering/formatting an XML tree
 - ✦ XPath



```

<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="Company.xsl"?>
<company>
  <dept code="software">
    <emp role="manager">
      <name>
        <first>Fred</first>
        <last>Flintstone</last>
      </name>
      <salary>80000.50</salary>
    </emp>
    <emp role="programmer">
      <name>
        <first>Joe</first>
        <last>Hacker</last>
      </name>
      <salary>42000.00</salary>
    </emp>
  </dept>
  <dept code="strategy">
    <emp role="manager">
      <name>
        <first>Poin</first>
        <middle>T</middle>
        <last>Hair</last>
      </name>
      <salary>100000.00</salary>
    </emp>
    <emp role="artist">
      <name>
        <first>Leonardo</first>
        <last>Bloggs</last>
      </name>
      <salary>13000.99</salary>
    </emp>
    <emp role="secretary">
      <name>
        <first>Felicity</first>
        <last>Typewell</last>
      </name>
      <salary>22000.10</salary>
    </emp>
  </dept>
</company>

```



```

<?xml version = "1.0" ?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">
  <xsl:template match="/">
    <html>
      <body>
        <xsl:for-each select="company/dept">
          <p>
            Department:
            <xsl:apply-templates select="@code" />
          </p>
          <p>
            Manager:
            <xsl:apply-templates
              select="emp[@role='manager']/name" />
          </p>
          <p>Others:</p>
          <ul>
            <xsl:for-each select="emp[@role!='manager']">
              <li>Employee (<xsl:value-of select="@role"/>):
                <xsl:apply-templates select="name" /></li>
            </xsl:for-each>
          </ul>
        </xsl:for-each>
      </body>
    </html>
  </xsl:template>
  <xsl:template match="name">
    <xsl:apply-templates select="first" />
    <xsl:apply-templates select="middle" />
    <strong><xsl:apply-templates select="last" /></strong>
  <xsl:define-template-set>
    <xsl:template match="first">
      <xsl:value-of />
    </xsl:template>
    <xsl:template match="middle">
      <xsl:value-of />.
    </xsl:template>
    <xsl:template match="last">
      <xsl:value-of />
    </xsl:template>
  </xsl:define-template-set>
  </xsl:template>
  <xsl:template match="@code">
    <em><xsl:value-of /></em>
  </xsl:template>
</xsl:stylesheet>

```

The background is a dark blue gradient with several large, semi-transparent gear shapes scattered across it. On the left side, there is a vertical strip containing a collage of various gears in different colors and sizes, including orange, yellow, and grey.

Programming XML

Two Styles of Processing

- ☀ two different styles of processing

- ☀ Document Object Model (DOM)

- W3C standard

- level 1 recommended in October 1998
- work on level 2 underway

- based around the notion of a document tree

- flexible; resource-hungry; fiddly

- actually two sets of interfaces

- core
- HTML

- ☀ Simple API for XML (SAX)

- an open-source, community developed system

- released May 1998
- widely used

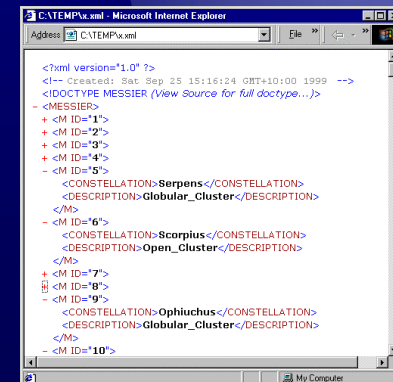
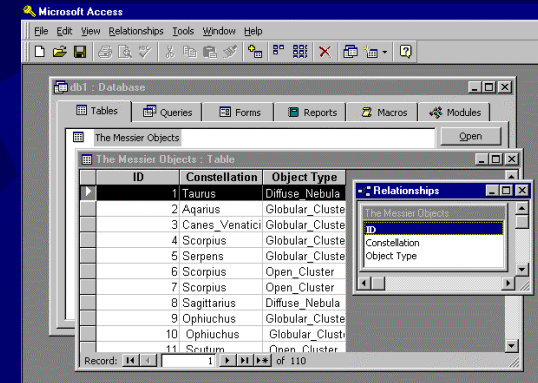
- sees a document as a stream of events that may be handled as needed

- efficient; simple; fast

Processing the DOM

```
import java.sql.*;
import com.datachannel.xml.om.*;

public class SQLDumper
{
    static
    { try { Class.forName ("com.ms.jdbc.odbc.JdbcOdbcDriver"); } catch (Exception e) {} }
    public static void main (String [] args) throws Exception
    {
        Document doc = new Document ();
        doc.appendChild (doc.createProcessingInstruction ("xml", "version=\"1.0\""));
        doc.appendChild (doc.createComment ("Created: " + new java.util.Date ());
        IXMLDOMELEMENT root = (IXMLDOMELEMENT) doc.createElement ("MESSIER");
        DocumentType.createDocumentType (doc, "MESSIER SYSTEM \"MessierDTD.dtd\"");
        doc.appendChild (root);
        Statement statement =
            DriverManager.getConnection ("jdbc:odbc:The Messier Database",
                                       "Messier", "MessierMan").createStatement ();
        ResultSet rs = statement.executeQuery ("SELECT * FROM [The Messier Objects]");
        while (rs.next ())
        {
            int id = rs.getInt ("ID");
            IXMLDOMELEMENT child = (IXMLDOMELEMENT) doc.createElement ("M");
            child.setAttribute ("ID", "" + id);
            IXMLDOMELEMENT c = (IXMLDOMELEMENT) doc.createElement ("CONSTELLATION");
            c.appendChild (doc.createTextNode (rs.getString ("Constellation")));
            child.appendChild (c);
            IXMLDOMELEMENT d = (IXMLDOMELEMENT) doc.createElement ("DESCRIPTION");
            d.appendChild (doc.createTextNode (rs.getString ("Object Type")));
            child.appendChild (d);
            root.appendChild (child);
        }
        System.out.println (doc.getXML ());
    }
}
```



Processing with SAX

```
import org.xml.sax.*;
import org.xml.sax.helpers.ParserFactory;

class MessierHandler extends HandlerBase
{
    private static final String descTagName = "DESCRIPTION";
    private int nGalaxies = 0;
    private boolean inDescriptionElement = false;
    public void endDocument ()
    {
        System.out.println ("There are " + nGalaxies + " Messier galaxies.");
    }
    public void startElement (String name, AttributeList atts)
    {
        inDescriptionElement = name.equals (descTagName);
    }
    public void endElement(String name)
    {
        inDescriptionElement = ! name.equals (descTagName);
    }
    public void characters (char ch [], int start, int length)
    {
        if (inDescriptionElement && new String (ch, start, length).endsWith ("alaxy"))
            nGalaxies ++;
    }
}

public class SAXCount
{
    public static void main (String [] args) throws Exception
    {
        Parser parser = ParserFactory.makeParser ("com.ibm.xml.parsers.SAXParser");
        parser.setDocumentHandler (new MessierHandler ());
        parser.parse (args [0]);
    }
}
```

```
bash "C:\TEMP\SQL Db"
bash-2.02$ jview SAXCount x.xml
There are 39 Messier galaxies.
bash-2.02$
```



XML & the Browser

Internet Explorer

- ★ MS came to the XML plate fairly quickly
 - ★ supports much of what we have looked at already:
 - CSS level 1/some level 2 (“spotty support”)
 - DOM level 1
 - XSL
 - Schemas
 - Namespaces
 - CDF
 - VML
 - XML Data Islands
 - XML embedded in HTML
 - also some ‘cool’ supporting features
 - HTML+TIME
 - XPath
 - etc.
 - ★ many of these are “technology previews”
 - ★ changeable; not standards compliant
 - but at least there is something to tinker with...
 - ★ MS is upgrading support all the time


```

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML//EN">
<html>
  <head>
    <title>NamedNodeMap Processing</title>
    <xml id="CD" src="CD.xml"></xml>
    <script for=window event=onload>
      <!--
      div.innerHTML = "";
      var xmlDoc = document.all("CD").XMLDocument;
      var root = xmlDoc.documentElement
      var attrs = root.attributes;
      div.innerHTML += "<strong>Title: </strong>" +
        attrs.getNamedItem("title").text + "<br>";
      div.innerHTML += "<strong>Artist: </strong>" +
        attrs.getNamedItem("artist").text + "<br>";
      div.innerHTML += "<strong>Date: </strong>" +
        attrs.getNamedItem("date").text + "<br>";
      div.innerHTML += "<strong>Number of tracks: </strong>" +
        attrs.getNamedItem("ntracks").text + "<br>";

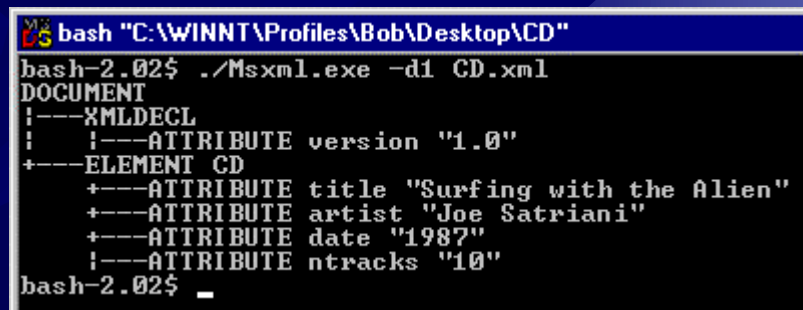
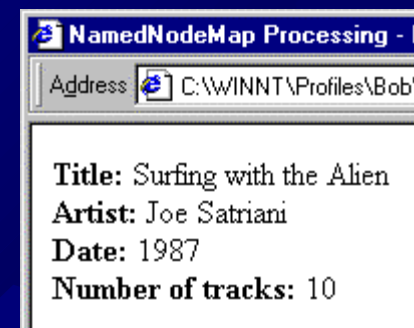
      -->
    </script>
  </head>
  <body>
    <p>
      <DIV ID="div"></DIV>
    </p>
  </body>
</html>

```

```

<?xml version="1.0"?>
<CD title="Surfing with the Alien"
  artist="Joe Satriani"
  date="1987"
  ntracks="10" />

```



Netscape Navigator

- ☀ nothing for Navigator 4...all XML work carried out in Mozilla
 - ✿ good support for CSS
 - ✿ DOM level 1
 - ✿ RDF support
 - XUL (eXtensible User interface Library)
 - ✿ simple XLink
 - permits *transclusion*...
 - embedded documents



XML & the Server

ASP

- ★ XML is not just a browser-based tool; also valuable at the server side
 - ★ makes it possible to deploy content in XML and transform it to HTML on demand for “down-level” clients
 - ★ will probably be one of the major uses

```
<%@ LANGUAGE = "JScript" %>
<%
    // Set the source and style sheet locations here
    var sourceFile = Server.MapPath("simple.xml");
    var styleFile = Server.MapPath("simple.xsl");

    // Load the XML
    var source=
        Server.CreateObject("Microsoft.XMLDOM");
    source.async = false;
    source.load(sourceFile);

    // Load the XSL
    var style =
        Server.CreateObject("Microsoft.XMLDOM");
    style.async = false;
    style.load(styleFile);

    // do the transformation
    Response.Write(source.transformNode(style));
%>
```



XML Ecology

Industry Initiatives

✦ bringing XML further into vertical and horizontal markets by providing tools & frameworks, schema distribution mechanisms, steering organisations, etc.

- ✦ Schema.net

- ✦ RosettaNet

- ✦ Biztalk.org

 - Microsoft's "No Glue" initiative

- ✦ CommerceNet

- ✦ OASIS

The background is a dark blue gradient with several large, semi-transparent gear shapes scattered across it. On the left side, there is a vertical strip of a colorful, textured image featuring various gears in shades of orange, yellow, and brown.

Gallery

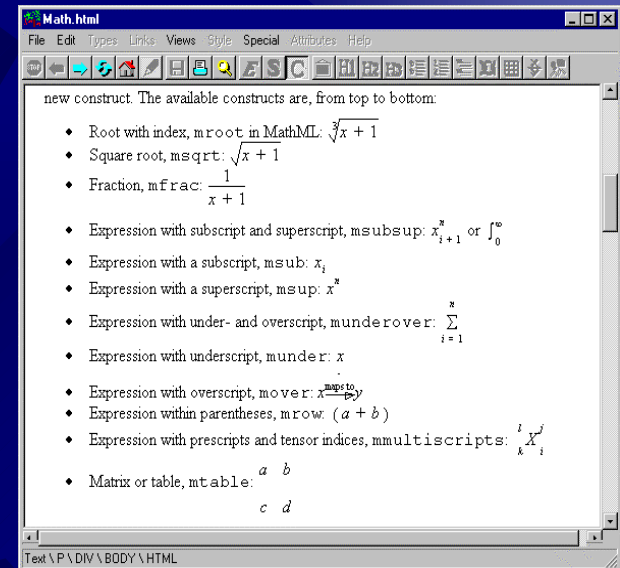
MathML

☀ supporting equations was part of the *original* goal of the World-Wide Web

```
<math>
  <mrow>
    <mi>y</mi>
    <mo>=</mo>
    <mi></mi>
    <mfrac href="http://www.w3.org/" xml:link="simple">
      <mn>1</mn>
      <msqrt>
        <mrow>
          <msup>
            <mi>x</mi>
            <mn>2</mn>
          </msup>
          <mo>+</mo>
          <mn>1</mn>
        </mrow>
      </msqrt>
    </mfrac>
  </mrow>
</math>
```

$$y = \frac{1}{\sqrt{x^2 + 1}}$$

“...MathML has a dual purpose: to provide a standard for mathematics on the Web and to provide a mathematical notation, which encapsulates the content of the mathematics as much as possible. The equations can then be used where they sit in a document or can be pulled from a document to be used in an entirely different application.”



Two MathML Modes

★ presentation markup

★ making marked-up data look nice

- 28 MathML presentation elements, with about 50 attributes

★ content markup

★ capturing the 'meaning' of the equation in a form suitable for automated processing

- around 75 content markup elements, with about a dozen attributes

Scalable Vector Graphics

- ★ a vocabulary for describing two-dimensional graphics in XML
- intended to replace bitmapped graphics
- much work is being done on producing SVG tools
 - Adobe, Corel, etc. producing export filters and browser plug-ins
 - Microsoft also promotes its own format, VML...

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG December 1999//EN"
  "http://www.w3.org/Graphics/SVG/SVG-19991203.dtd">
<svg width="12cm" height="4cm" viewBox="0 0 1200 400">
  <desc>Example rect02 - rounded rectangles expressed in user coordinates</desc>

  <rect x="100" y="100" width="400" height="200" rx="50"
    style="fill:green;" />

  <g transform="translate(700 300); rotate(-30)">
    <rect x="0" y="0" width="400" height="200" rx="50"
      style="fill:none; stroke:purple; stroke-width:30" />
  </g>
</svg>
```



SMIL

- ★ Synchronised Multimedia Integration Language
 - ★ a vocabulary for the creation of multimedia presentations
 - W3C recommendation; not widely adopted
 - Quicktime 4.1 and RealPlayer G3 are the only common applications
 - ★ SMIL documents are ‘glue’ that tell a player application which resources to retrieve and when they should be presented
 - relies on stylesheets for formatting, etc.
 - ★ metadata ‘switch’ element allows SMIL software to adapt to different platform/bandwidth conditions
 - ★ perceived to be isolated/too ‘academic’
 - doesn’t fit in well with DHTML, etc.

```
<smil>
  <head>
    <layout>
      <root-layout width="640" height="480" background-color="black" />
      <region id="logo" left="20" top="5" width="100" height="50" />
      <region id="vidbk" left="200" top="50" width="150" height="76"
        background-color="#330033" z-index="1" />
      <region id="video" left="210" top="55" width="100" height="70"
        background-color="#000000" z-index="3" />
      <region id="ccbkc" left="20" top="200" width="400" height="30"
        background-color="#666600" z-index="2" />
      <region id="ccscroll" left="21" top="210" width="350" height="25"
        fit="fill" z-index="2" />
    </layout>
  </head>
  <body>
    <par>
      <seq>
        <par>
          
        </par>
        <par>
          <video src="video.avi" region="logo" fill="freeze" />
          <text src="cctext.txt" region="ccscroll" fill="freeze" />
        </par>
      </seq>
    </par>
  </body>
</smil>
```

HTML+TIME

- ✦ proposed to W3C by Microsoft, Macromedia & Compaq
 - ✦ attempt to overcome SMIL's perceived shortcomings with respect to browsers
 - ✦ allows time attributes to be applied to any element
- ✦ supported in IE5

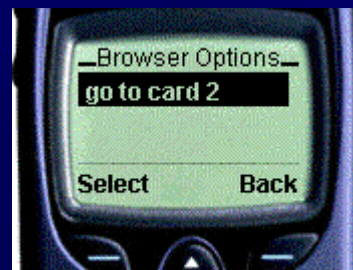
```
<HTML>
<HEAD>
<STYLE TYPE="text/css">
    .time      { behavior:url(#default#time); }
</STYLE>
<XML:NAMESPACE PREFIX="t"/>
</HEAD>
<BODY>
<DIV CLASS="time" t:REPEAT="3" t:DUR="8" t:TIMELINE="par">
    <DIV CLASS="time" t:BEGIN="0" t:DUR="4">First line of text.</DIV>
    <DIV CLASS="time" t:BEGIN="2" t:DUR="4">Second line of text.</DIV>
    <DIV CLASS="time" t:BEGIN="4" t:DUR="4">Third line of text.</DIV>
    <DIV CLASS="time" t:BEGIN="6" t:DUR="4">Fourth line of text.</DIV>
</DIV>
</BODY>
</HTML>
```

WML Example

- ★ a key part of WAP
- ★ allows for the presentation and delivery of data and telephony services on mobile wireless terminals
- ★ a WML document is composed of a deck, and a deck contains multiple cards

```
<?xml version="1.0"?>
<!DOCTYPE WML PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
  "http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>
  <card id="card1" title="Example 1">
    <!-- a card can only contain P or DO blocks -->
    <p>
      <do type="accept" label="go to card 2">
        <go href="#card2"/>
      </do>
      This is the first card.
    </p>
  </card>

  <card id="card2" title="Example 1">
    <p>
      This is the second card.
    </p>
  </card>
</wml>
```



SOAP

- ★ Object-Object Protocols growing in popularity
 - ★ DCOM, CORBA, RMI, etc.
 - ★ good in an intranet situation; bad for the internet: require gaping holes to be left in an organisation's firewall
- ★ SOAP aims to allow any OO protocol to 'tunnel' through port 80
 - ★ used for the World-Wide Web and already open at many sites
- ★ uses XML to define the format of request and response messages and then allows the use of the normal HTTP POST command to send this information
 - ★ *"What is SOAP if not basically a more object-oriented, somewhat buzzword-compliant upgrade to CGI?"*


```
boolean PlaceOrder([in] Title string,  
                  [in] Author string,  
                  [out] DaysToDelivery integer);
```

```
POST /BookServer HTTP/1.1  
Host: www.qwickbooks.com  
Content-Type: text/xml-SOAP  
Content-Length: nnnn  
SOAPMethodName: Some-Namespace-URI#PlaceOrder
```

```
<SOAP:Envelope xmlns:SOAP="urn:schemas-xmlsoap-org:soap.v1">  
  <SOAP:Body>  
    <m:PlaceOrder xmlns:m="Some-Namespace-URI">  
      <Title>Happy All The Time</Title>  
      <Author>Laurie Colwin</Author>  
    </m:PlaceOrder>  
  </SOAP:Body>  
</SOAP:Envelope>
```



```
HTTP/1.1 200 OK  
Connection: close  
Content-Type: text/xml  
Content-Length: nnnn
```



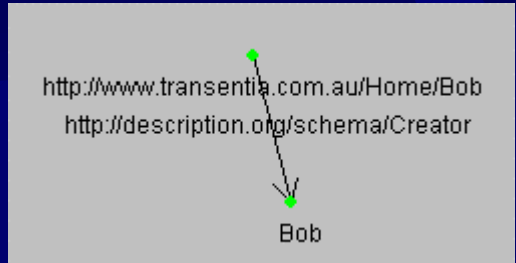
```
<SOAP:Envelope xmlns:SOAP="urn:schemas-xmlsoap-org:soap.v1">  
  <SOAP:Body>  
    <m:PlaceOrderResponse xmlns:m="Some-Namespace-URI">  
      <return>1</return>  
      <DaysToDelivery>7</DaysToDelivery>  
    </m:PlaceOrderResponse>  
  </SOAP:Body>  
</SOAP:Envelope>
```


RDF

★ Resource Description Framework:

- ★ a foundation for processing metadata
 - data about data
- ★ provides facilities to enable automated processing of Web resources
- ★ useable in a variety of application areas:
 - resource discovery to enhance search engines
 - for describing the content and content relationships available at a particular Web site, or digital library
 - in describing collections of pages that represent a single logical “document”
 - by intelligent software agents to facilitate knowledge sharing and exchange
 - in content rating
 - RDF plus digital signatures will be key to building the “Web of Trust” for electronic commerce, collaboration, and other applications

<http://jigsaw.w3.org:8000/description>



“Bob is the creator of the resource http://www.transentia.com.au/Home/Bob”

object predicate subject


```
<?xml version="1.0"?>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:s="http://description.org/schema/">
  <rdf:Description about="http://www.transentia.com.au/Home/Bob">
    <s:Creator>Bob</s:Creator>
  </rdf:Description>
</rdf:RDF>
```

The background is a dark blue field filled with various sizes of gear silhouettes in shades of blue and black. On the left side, there is a vertical strip of colorful, textured gears in shades of orange, yellow, and brown.

Soothsaying....

To be Successful...

- ✦ people and organisations must see the value of XML
 - ✦ as they did with HTML
 - but now we need more than “pretty pictures”...
- ✦ better tools must become available
 - ✦ *“...the web browser must become a stable building block for site designers, just as standardisation on Windows has encouraged innovation in the PC space.”*
 - ✦ *“We have to live in the present, and therein lies the problem with XML and browsers.”*
- ✦ standardization needs to continue
 - ✦ and then be adopted properly

The background features a dark blue field with several large, semi-transparent gears of various sizes. On the left side, there is a vertical strip with a colorful, abstract, and textured appearance, possibly representing a gear mechanism or a data visualization. The text is presented in a white, italicized serif font.

"It's evident that XML is finding its way into every facet of the software industry. It's becoming an integral part of database technologies (such as DBMS and ADO), remote procedure call mechanisms such as SOAP, and business-to-business integration and messaging software such as BizTalk™. XML is showing up in Web browsers and servers such as Internet Explorer 5.0 and Internet Information Services 5.0, and many other domain-specific applications."

"The XML family of standards will emerge as the dominant technical foundation by year-end 2000, continuing support of pre-existing HTML documents, but used in the creation of new documents and document applications (0.6 probability)."



XML Resources

XML Resources

- ★ The W3C's definitive site for XML
<http://www.w3c.org/XML>
- ★ Microsoft's XML site <http://msdn.microsoft.com/xml>
- ★ The XML Cover page
<http://xml.com/xml/pub/coverpage/newspage.html>
- ★ XMLSoftware
<http://xmlsoftware.com>