

CM0133 Internet Computing

XML

The eXtensible Markup Language

- XML and HTML
- XML applications
- XML documents and the XML data model
- XML applications
 - Documents
 - Type Declarations and Definitions
 - Stylesheets

XML and HTML

- HTML elements describe the **structure** of a document and the **style** of presentation
 - HTML elements do not indicate the **meaning** of the information contained in the document
- XML allows authors to create their own tags (elements)
 - tags can be used to describe the **meaning** of the information contained within them (i.e. within the element)
 - we can also define **attributes** for these tags
- XML documents represent the **structure** of the information
 - by allowing a hierarchical ordering of the elements
- Scripts can make sophisticated use of XML tags
 - for example, to display the information on a web browser

XML and HTML

```
<ul>
  <li>Web Programming</li>
  <li>Chris Bates</li>
  <li>John Wiley and Sons</li>
  <li>2002</li>
  <li>0-470-84371-3</li>
</ul>
```

```
<book type="technical">
  <title>Web Programming</title>
  <author>Chris Bates</author>
  <publisher>John Wiley and Sons</publisher>
  <year>2002</year>
  <ISBN>0-470-84371-3</ISBN>
</book>
```

- NOTE: XML does not DO anything!
 - Created to structure, store and send information
 - HTML designed to DISPLAY data
- Why XML?
 - On internet, XML describe data, HTML display data
 - Can have multiple views of same data
 - Exchange data between incompatible systems/different platforms
 - Just exchange information in plain text files
 - B2B (Business to Business)
- Future applications all likely to exchange data in XML

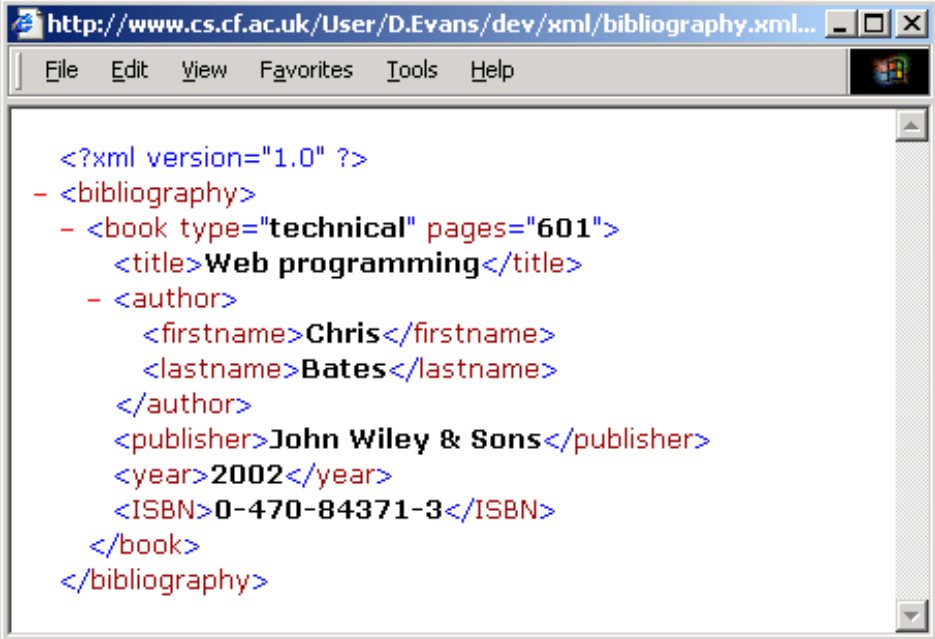
- XML is a **meta-language** (a subset of SGML)
 - used to create custom markup languages
 - provides a basic format for structured documents
- XML allows authors to define their own elements
 - used to describe the **meaning** of the information they contain
 - we identify different types of information according to the meaning of that information
- There is no standard set of XML tags, but many widely-used markup languages have been created using XML
 - CML (chemical markup language)
 - MathML (mathematical markup language)
 - MusicML (musical markup language)

A simple XML Document

```
<?xml version="1.0"?>
<bibliography>
  <book type="technical" pages="601">
    <title>Web programming</title>
    <author>
      <firstname>Chris</firstname>
      <lastname>Bates</lastname>
    </author>
    <publisher>John Wiley and Sons</publisher>
    <year>2002</year>
    <ISBN>0-470-84371-3</ISBN>
  </book>
</bibliography>
```

A simple XML Document

- The file is called **bibliography.xml**
- The first line is a **processing instruction** which specifies the XML version used
- The **bibliography** element is composed of one or more **book** elements
- The **book** element is has child elements **title**, **author**, **publisher**, **year** and **ISBN**
- The **author** element has child elements **firstname** and **lastname**
- The **book** element has attributes **type** and **pages**

A screenshot of a web browser window showing an XML document. The address bar displays the URL: http://www.cs.cf.ac.uk/User/D.Evans/dev/xml/bibliography.xml... The browser's menu bar includes File, Edit, View, Favorites, Tools, and Help. The main content area shows the following XML code:

```
<?xml version="1.0" ?>
- <bibliography>
  - <book type="technical" pages="601">
    <title>Web programming</title>
    - <author>
      <firstname>Chris</firstname>
      <lastname>Bates</lastname>
    </author>
    <publisher>John Wiley & Sons</publisher>
    <year>2002</year>
    <ISBN>0-470-84371-3</ISBN>
  </book>
</bibliography>
```


CML example

```
<?xml version="1.0"?>
<cml xmlns="http://www.xml-cml.org/schema/cml2/core">
  <molecule id="myMolecule">
    <atomArray>
      <atom id="a1" elementType="C" hydrogenCount="0"/>
      <atom id="a2" elementType="C" hydrogenCount="0"/>
      <atom id="a3" elementType="C" hydrogenCount="2"/>
    </atomArray>
    <bondArray>
      <bond atomRefs="a1 a2" order="1"/>
      <bond atomRefs="a2 a3" order="1"/>
      <bond atomRefs="a1 a3" order="2"/>
        <stereo>W</stereo>
      </bond>
    </bondArray>
  </molecule>
</cml>
```

MathML example

```
<?xml version="1.0"?>
<math xmlns="http://www.w3.org/1998/Math/MathML">
  <mstyle fontsize="30pt">
    <mrow>
      <msup>
        <mi>x</mi>
        <mn>2</mn>
      </msup>
      <mo>+</mo>
      <mrow>
        <mn>4</mn>
        <mo>&InvisibleTimes;</mo>
        <mi>x</mi>
      </mrow>
    </mrow>
  </mstyle>
</math>
```

- XML is **case sensitive**
- An XML document must be **well-formed**
 - every opening tag must have a closing tag
 - elements must not overlap
 - all attribute values must be enclosed in quotation marks (single or double)
- XML documents are often required to obey certain rules regarding the structure of their elements
 - these rules are specified in a **document type declaration**
 - this leads to the concept of **valid** XML documents

Well-formed XML documents

- The document must have one element (the root) within which all other elements are nested
- All attribute values must be in quotation marks
- All elements must have opening and closing tags, unless empty in which case `<tagname/>` must be used
- All tags must be properly nested
 - opening and closing tags must be inside their parent
- Markup characters must not be used in document text
 - `<, >, &,]]>`
- Entities must be declared in a DTD

Pair up and write a well formed XML document for describing cars

Examples

```
<!DOCTYPE node PUBLIC "-//freedesktop//DTD
D-BUS Object Introspection 1.0//EN"
```

```
"http://www.freedesktop.org/standards/dbus/1.0/i
ntrospect.dtd">
```

```
<node name="/com/trolltech/examples/car">
```

```
  <interface
name="com.trolltech.Examples.CarInterface">
```

```
    <method name="accelerate"/>
```

```
    <method name="decelerate"/>
```

```
    <method name="turnLeft"/>
```

```
    <method name="turnRight"/>
```

```
    <signal name="crashed"/>
```

```
  </interface>
```

```
</node>
```

```
class XmlExamples {
  static def CAR_RECORDS = ""
    <records>
      <car name='HSV Maloo' make='Holden' year='2006'>
        <country>Australia</country>
        <record type='speed'>Production Pickup Truck with speed of
271kph</record>
      </car>
      <car name='P50' make='Peel' year='1962'>
        <country>Isle of Man</country>
        <record type='size'>Smallest Street-Legal Car at 99cm wide and 59
kg in weight</record>
      </car>
      <car name='Royale' make='Bugatti' year='1931'>
        <country>France</country>
        <record type='price'>Most Valuable Car at $15 million</record>
      </car>
    </records>
  ""
}
```

Valid XML documents

- The document must be well formed
- The document's root element must match the root element specified in the associated DTD
- The document must have a DTD that declares all elements, attributes and entities
- The document must follow the rules (grammar) specified in the associated DTD

XML Parsers

- HTML
 - If errors in HTML then still works
 - Leads to different browsers interpreting HTML slightly differently
 - Leads to incompatibility issues between browsers
- XML
 - Decided this should not be the case. If error in XML, then program should not continue
 - XML parsers created to check well-formed XML

XML Parsers

- XML parsers construct a **tree representation** of the data
 - The majority of XML parsers are non-validating
 - They only check that the document is well-formed
- Browser includes an XML parser
- Other XML parsers:
 - SAX-based parsers
 - DOM-based parsers

XML applications

An XML application has three components

- An **XML document**
 - contains data tagged with content-specific elements
 - There is no standard set of XML tags.
- A **document type definition (DTD)**
 - specifies element names and attributes, and rules for the hierarchical structuring of elements.
 - There are various specifications of tags, defined in DTDs that may be public or private
- A **stylesheet**
 - specifies formatting rules for the document
 - either CSS (cascading stylesheet) or XSL (Extensible Stylesheet Language)

XML documents

- An XML document is described by a **data model**
- The data model is a **tree** consisting of
 - Element nodes
 - Control Nodes
 - Document Nodes
 - Processing instruction nodes
 - Comment nodes
 - Data nodes

Element nodes

An **element node** is created by an expression like

```
<eltType a1="A1" ... an="An">c1 . . . cm</eltType>
```

or

```
<eltType a1="A1" ... an="An"/>
```

- Each element node has
 - An **element type**: **eltType** (this is the tag name)
 - A **set of attribute-value pairs**: { (a_i, "A_i") }
 - An **ordered list of children**: {c_j}
- Note: each attribute a_i must be unique

Elements

- are used to tag the various components that comprise the logical structure of a document
- are defined in a document type definition
 - this is accessed using a document type declaration
- may contain other elements and may include attributes
- may be empty, as in `<tagName/>`

Document nodes

A **document node** is a particular kind of element node

```
<!doctype eltType "URL">c1 . . . cm
```

- A document node has a **type** but no attributes. Instead, it has an optional URL which specifies a **data model** for this node and its children.
- Exactly one child of a document node must be an element node (of the same type as the document type)
- The root node of the XML tree may be an **anonymous document node** (without a type and without a URL)
 - Such document nodes are represented by the **absence** of a **<!doctype>** element

Document type declarations

- A **document type declaration** is a single document node which defines a data model for the entire document

```
<!DOCTYPE bibliography SYSTEM "myBib.dtd">
```

- Specifies the location of a **document type definition**
 - In this case, the file “**myBib.dtd**”
 - The child node of the DTD is the root element of the document
 - The DTD could also be included in the XML document itself
- **SYSTEM** indicates that the file is on a local computer
 - **PUBLIC** would indicate that the DTD is publicly available
- Specifies the **root element** of the document
 - **bibliography** is the root element

Processing instruction nodes

A **processing instruction** node is always a leaf node, and only has a processing instruction associated with it

```
<? a processing instruction ?>
```

- A processing instruction is any sequence of characters, the only restriction being that the sequence may not start with the three characters **xml** (upper, lower or mixed case) followed by a space or newline.
- Instructions starting with **xml** followed by a whitespace character have special meaning.

```
<?xml a special processing instruction ?>
```


Processing instruction nodes

Processing instruction nodes contain information that can be used by application programs

- processing instructions are ignored by XML parsers

1) The following line is mandatory (specifies xml version)

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

2) The following declares that external files are required

```
<?xml version="1.0" standalone="no" ?>
```

3) The following includes a reference to an XSL stylesheet

```
<?xml-stylesheet href="mysty.xsl" type="text/xsl" ?>
```

Comment nodes

A **comment node** is similar to a processing instruction node – it is always a leaf node and contains only a comment

```
<!-- a comment -->
```

- Comment nodes are used to include **explanatory notes** for human consumption
- Processing instruction nodes are for **consumption by an application**
- In the XML data model there is no difference between processing instruction nodes and comment nodes

Data nodes

- A **data node** is always a leaf node and has only a single characteristic – the data itself

```
<aTag>  
    some data  
</aTag>
```

- Since all the other types of nodes have delimiters that distinguish them, data nodes don't need delimiters
 - Everything not contained between “<” and “>” is data
- Data nodes cannot be empty
 - their data characteristic must contain at least one character

Example XML document

```
<?xml version="1.0"?>
<!DOCTYPE bibliography SYSTEM "myBib.dtd">
<!-- This is my bibliography -->
<bibliography>
  <book type="technical" pages="601">
    <title>Web programming</title>
    <author>
      <firstname>Chris</firstname>
      <lastname>Bates</lastname>
    </author>
    <publisher>John Wiley and Sons</publisher>
    <year>2002</year>
    <ISBN>0-470-84371-3</ISBN>
  </book>
</bibliography>
```

Document Type Definitions (DTD)

- An XML document has neither meaning nor context without a **grammar** against which it can be validated
- The grammar is called a **Document Type Definition**
- Writing a good DTD is probably the most difficult aspect of writing an XML application
- The DTD has only a few components
 - The way that these components are assembled leads to complex structures (like the bibliography)
 - A DTD is primarily used to verify XML documents. Good practice in business etc.

Example DTD

```
<!ELEMENT bibliography (book+) >
  <!ATTLIST bibliography
    title CDATA "Bibliography">
  <!ELEMENT book (title, author+, publisher, year, ISBN)>
    <!ATTLIST book
      type (technical | biography | fiction) #REQUIRED
      pages CDATA #IMPLIED >
    <!ELEMENT title (#PCDATA)>
    <!ELEMENT author (firstname, initial*, lastname)>
      <!ELEMENT firstname (#PCDATA)>
      <!ELEMENT initial (#PCDATA)>
      <!ELEMENT lastname (#PCDATA)>
    <!ELEMENT publisher (#PCDATA)>
    <!ELEMENT year (#PCDATA)>
    <!ELEMENT ISBN (#PCDATA)>
  <!ENTITY isbn "ISBN:">
```

Example DTD

- The **bibliography** element is the **root element** of the DTD, and contains one or more **book** elements
 - **book** exactly one occurrence
 - **book?** zero or one occurrence
 - **book+** one or more occurrences
 - **book*** zero or more occurrence
- The **book** element contains 5 child elements: **title**, **author+**, **publisher**, **year** and **ISBN**
 - these must be included in the specified order
- **(title|author+|publisher|year|ISBN)**
 - indicates that any ordering is acceptable

Example DTD

- The book element has two attributes: **type** and **pages**
 - **PCDATA**
 - indicates that the data should be parsed (by the parser)
 - data can only contain “legal” characters and defined entities
 - **CDATA**
 - indicates that the data should be ignored by the parser
 - the data can contain any characters
 - **#REQUIRED** means mandatory (must be present)
 - **#IMPLIED** means optional
- **type (technical|biography|fiction) #REQUIRED**
 - The value of the **type** attribute must be either **technical**, **biography** or **fiction**

Example DTD

- Internal entities

```
<!ENTITY isbn "ISBN:">
```

- This defines an internal entity called `isbn`
- Internal entities are used to create small pieces of data that are to be used repeatedly throughout the document
- When an entity is included, its name is preceded by an ampersand (&) and followed by a semicolon(;).
- The entity reference `&isbn;` is replaced by the string “ISBN:”
- This is exactly the same way that HTML control characters are included in documents (e.g `<` for the `<` character)

- External entities

```
<!ENTITY myImage SYSTEM "myImage.png" NDATA PNG>
```

- This defines an external entity as a container for a PNG image

Cascading stylesheets

- **Recall:** XML does not contain display information
 - We invent tags. Therefore a browser doesn't know if e.g. <table> tag refers to HTML table or a dining table!
- **Different solutions to view problem: CSS, XSL, Javascript**
- **Cascading stylesheets** are a simple way to view XML applications on the web
- Cascading stylesheets are limited in what they can achieve – they have no support for tables or lists
- They are included using the following line:

```
<?xml-stylesheet type="text/css" href="myStyles.css"?>
```

Example

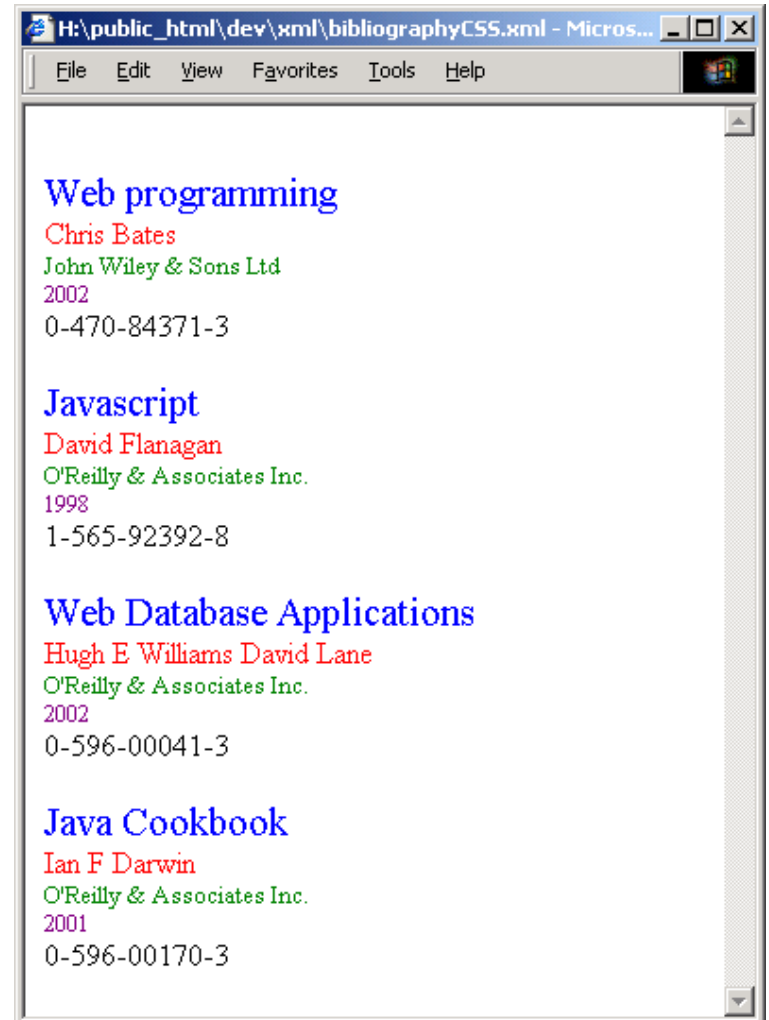
```
<?xml version="1.0"?>
<!DOCTYPE bibliography SYSTEM "myBib.dtd">
<?xml-stylesheet type="text/css" href="myStyles.css"?>

<bibliography name="Bibliography for CMT602c">
  <book type="technical" pages="601">
    <title>Web programming</title>
    <author>
      <firstname>Chris</firstname>
      <lastname>Bates</lastname>
    </author>
    <publisher>John Wiley & Sons Ltd</publisher>
    <year>2002</year>
    <ISBN>0-470-84371-3</ISBN>
  </book>
  ...etc...
</bibliography>
```

Cascading stylesheets

Part of `myStyles.css`

```
title {  
  font-family:"times";  
  font-size:16pt;  
  color:blue;  
  display:block;  
  padding-top:15pt;  
}  
... etc ...  
  
ISBN {  
  family:"times";  
  font-size:12pt;  
  color:black;  
  display:block;  
}
```



The Extensible Stylesheet Language

- A cascading stylesheet creates a style for specific XML elements
- An XSL stylesheet creates a **template** – this is a design for (part of) the page
- The template is used to format XML elements which match a specified pattern
- XSL can be used to produce **any** type of markup
 - HTML, LaTeX, PDF, Rich Text Format
- XSL stylesheets are included using the following line:

```
<?xml:stylesheet type="text/xsl" href="bibStyle.xsl"?>
```

Example

```
<html>
<body bgcolor="lightyellow">
  <h1><!-- put bibliography title here --></h1>
  <table border="1">

    <!-- for every book -->
    <tr>
      <td><!-- put title here --></td>
      <td><!-- put authors here --></td>
      <td><!-- put publisher here --></td>
      <td><!-- put year here --></td>
      <td><!-- put ISBN here --></td>
    </tr>
  </table>
</body>
</html>
```

- First write a framework for the desired output (using comments)

```

<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">
  <xsl:template match="/">
    <html>
      <body bgcolor="lightyellow">
        <h1><xsl:value-of select="bibliography/@name"/></h1>
        <table border="1">
          <xsl:for-each select="bibliography/book">
            <tr>
              <td><xsl:value-of select="title"/></td>
              <td>
                <xsl:for-each select="author">
                  <xsl:value-of select="firstname"/>
                  <xsl:value-of select="lastname"/><br/>
                </xsl:for-each>
              </td>
              <td><xsl:value-of select="publisher"/></td>
              <td><xsl:value-of select="year"/></td>
            </tr>
          </xsl:for-each>
        </table>
      </body>
    </html>
  </xsl:template>
</xsl:stylesheet>

```

Example

The screenshot shows a web browser window with the following content:

Bibliography for CMT602c			
Web programming	Chris Bates	John Wiley & Sons Ltd	2002
Javascript	David Flanagan	O'Reilly & Associates Inc.	1998
Web Database Applications	Hugh Williams David Lane	O'Reilly & Associates Inc.	2002
Java Cookbook	Ian Darwin	O'Reilly & Associates Inc.	2001

- The following line declares that the file is a stylesheet

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">
```

- The following line declares an XSL **template**

```
<xsl:template match="/">
```

- A stylesheet can contain multiple templates for use in different situations. This example defines a single template (which is applied to the whole document) using the pattern matching command **match**
- Any element matching the pattern will be subject to the transformations it includes

- The XML document is represented as a **hierarchy of patterns** (each separated by a forward slash)
- The following line iterates over all books

```
<xsl:for-each select="bibliography/book">
```

- The following line extracts the value of the book title

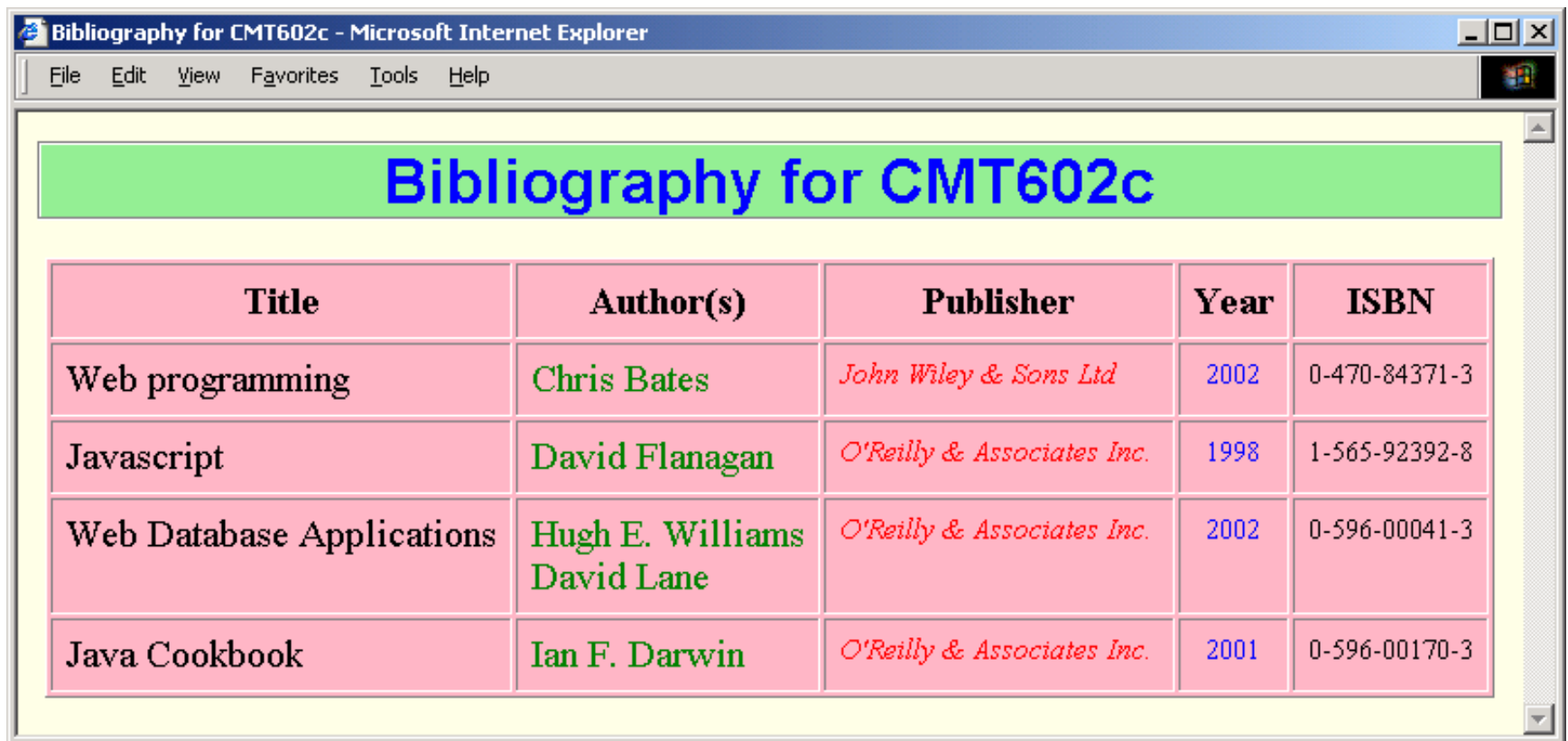
```
<xsl:value-of select="title"/>
```

- The tag is substituted in the output by the value
- The following line extracts the name attribute of the bibliography

```
<xsl:value-of select="bibliography/@name"/>
```

Example

- Using the attributes of the HTML elements (including style attributes) we can produce more complex presentations



The screenshot shows a Microsoft Internet Explorer browser window titled "Bibliography for CMT602c - Microsoft Internet Explorer". The browser's menu bar includes "File", "Edit", "View", "Favorites", "Tools", and "Help". The main content area features a green header bar with the text "Bibliography for CMT602c" in blue. Below the header is a table with five columns: "Title", "Author(s)", "Publisher", "Year", and "ISBN". The table contains four rows of data, each with a title, author(s), publisher, year, and ISBN.

Title	Author(s)	Publisher	Year	ISBN
Web programming	Chris Bates	<i>John Wiley & Sons Ltd</i>	2002	0-470-84371-3
Javascript	David Flanagan	<i>O'Reilly & Associates Inc.</i>	1998	1-565-92392-8
Web Database Applications	Hugh E. Williams David Lane	<i>O'Reilly & Associates Inc.</i>	2002	0-596-00041-3
Java Cookbook	Ian F. Darwin	<i>O'Reilly & Associates Inc.</i>	2001	0-596-00170-3

Summary

- XML and HTML
- XML applications
- XML documents and the XML data model
- XML applications
 - Documents
 - Type Declarations and Definitions
 - Stylesheets

<http://www.w3schools.com/xml/default.asp>