SWINBURNE UNIVERSITY OF TECHNOLOGY

COS10005
Web Development

Module 11 – Introduction to XML, AJAX and web services

Contents

- What is XML?
- XML Applications
- Reading XML Data
  - With JavaScript
- Web Requests
- Ajax

WHAT IS XML?

SWINBURNE UNIVERSITY OF TECHNOLOGY

What is XML ?

- XML is a simple ***structured general mark-up language***
- XML enables ***structured data*** to be marked-up, searched and utilized in ***XML Applications*** *… e.g., using the DOM* ☺
- XML data can be ***exchanged***:
  - between computers,
  - between computer applications,
  - between organizations.
- Electronic document ***data exchange*** is now easily arranged with XML and the Web, *e.g., using **Web Services** as the API*.

What is XML ?

- **Extensible Markup Language** (XML) is

  a human-readable,

  machine-understandable,

  general syntax for describing hierarchical data,

  applicable to a wide range of applications

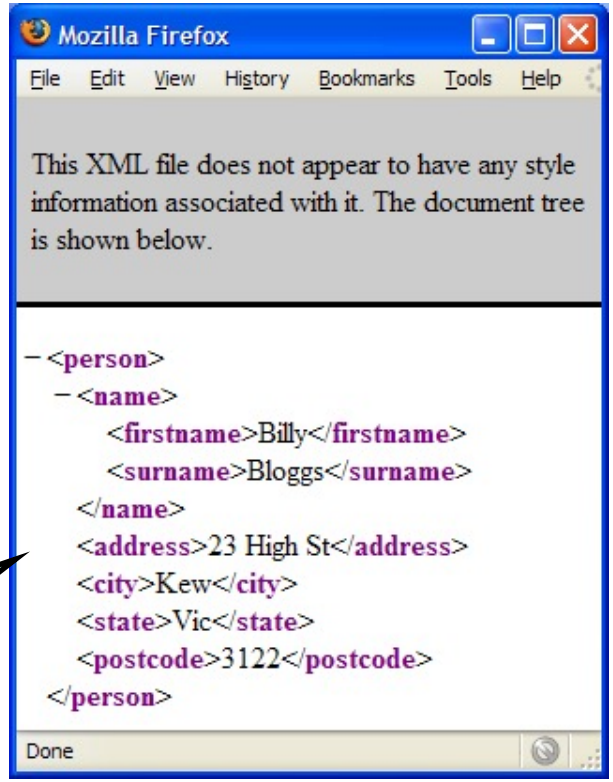
- XML is an ISO compliant ***subset*** of **Standard Generalized Markup Language** (SGML).
- XML (and SGML) is a **meta-language**
- XML is ***extensible***

XML

- A quick look:

Any structured data can be marked up with XML

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE person SYSTEM "person.dtd">
<person>
   <name>
        <firstname>Billy</firstname>
        <surname>Bloggs</surname>
   </name>
   <address>23 High St</address>
   <city>Kew</city>
   <state>Vic</state>
   <postcode>3122</postcode>
</person>
```



Most current browsers will render ***well-formed*** XML documents. ☺

XML Technologies

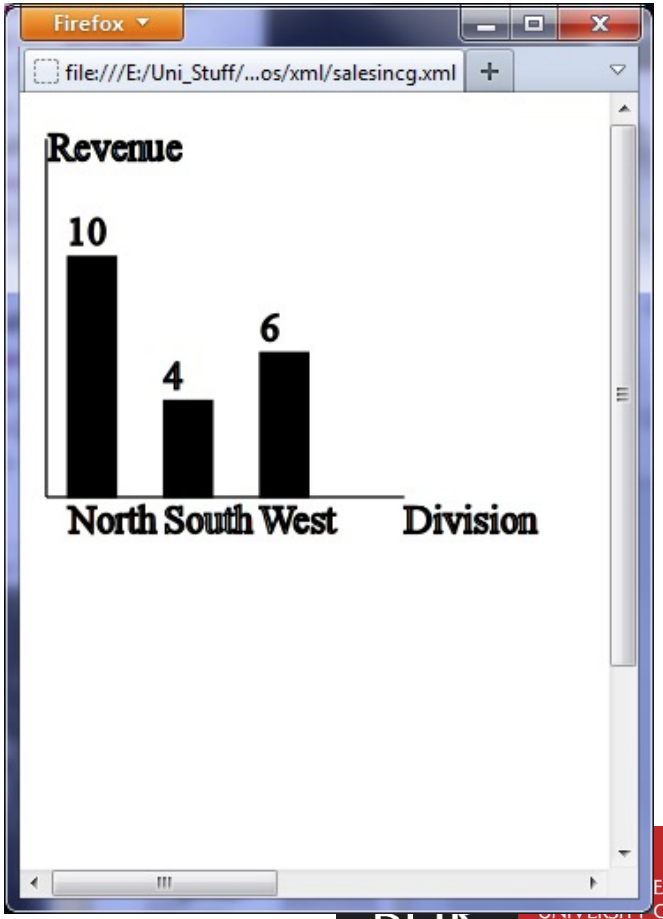
- XML is also a family of technologies
  - **XML** Syntax (Core) defines what “tags” and “attributes” are.
  - **XLink** defines how to add ***hyperlinks*** to an XML file.
  - **XPointer** defines how to ***point to parts*** of an XML file.
  - **XSL** (Extensible Style Sheet Language) can ***transform*** an XML
  - **XML Schema** used to ***define the structure*** on an XML.
  - **XML DOM** is used to access XML objects
- XML is extended and supported,
  by many associated technologies:
  such as **Document Type Definitions** (DTDs), **XML Namespaces**,
  XML Schema and **Resource Description Framework** (RDF).
- These technologies, and many more, are in varying stages of the W3C specification process, and adoption.

http://www.w3.org/XML/

XSL – e**X**tensible **S**tylesheet **L**anguage

- An XML document can be transformed into HTML, SVG, or PDF, etc, using XSL.

```
<?xml version="1.0" encoding="UTF-8"?>
<sales>
        <division id="North">
                <revenue>10</revenue>
                <growth>9</growth>
                <bonus>7</bonus>
        </division>

        <division id="South">

                <revenue>4</revenue>
                <growth>3</growth>
                <bonus>4</bonus>
        </division>

        <division id="West">
                <revenue>6</revenue>

                <growth>-1.5</growth>
                <bonus>2</bonus>
        </division>
</sales>
```



See http://www.w3.org/TR/xslt

XML Document

- Should contain a simple **version declaration** that tells the processor what version of XML the document conforms to:

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

- Is considered "**well-formed**" if it strictly follows the syntax requirements of XML
- Can be read by any XML-parser, if it is a well-formed XML document.

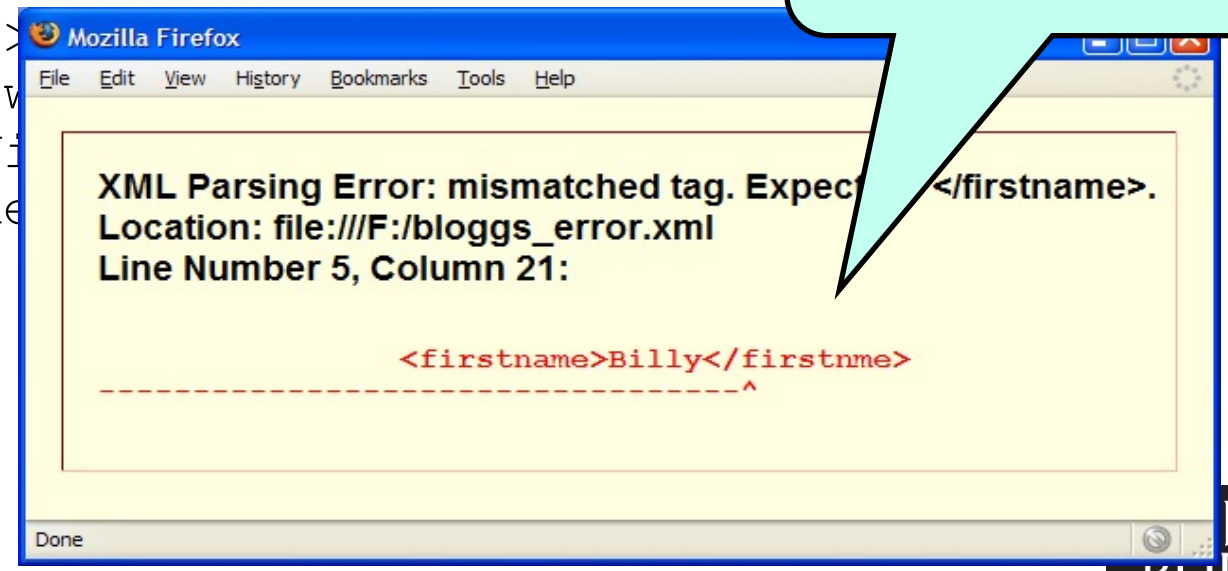
Well-formed XML

- Must have a single root element that encloses all the other elements.
- All elements must be properly nested within each other, with no overlapping or intersecting tags.
- All elements must have a start tag and an end tag, and the start and end tags must match.
- All attribute values must be enclosed in quotes.
- Special characters such as <, >, &, ', and " must be escaped using the corresponding character entities.

Well-Formed XML

- Not well-formed:

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE person SYSTEM "person.dtd">
<person>
   <name>
        <firstname>Billy</firstnme>
        <surname>Bloggs</surname>
   </name>
   <address>
   <city>Kew
   <state>Vi
   <postcode
</person>
```



Most browsers will **not** render XML documents that are **not** well-formed. ☹

What is “well-formed” ?

- **Rule 1:** All tags must be properly closed :
  - Incorrect: `<name>Billy Bloggs`
  - Correct:: `<name>Billy Bloggs</name>`
  - Correct: `<employee><name /></employee>`
- **Rule 2**: All tags must be properly nested:
  - Incorrect: `<employee><name> ... </employee ></name>`
  - Correct: `<employee><name> ... </name></employee>`
- **Rule 3**: All attribute values must be in double quotes.
  - Incorrect: `<price currency=AUD>`
  - Correct:: `<price currency="AUD">`

What is “well-formed” ?

- **Rule 4:** An element may not have two attributes with the same name.
  - Incorrect: `<price currency="AUD" currency="USD">`
  - Correct: `<price currency="AUD">`
- **Rule 5**: XML is case sentitive.
  - `<Atag>` `<atag>` , and `<ATAG>` are *three **different** tags*
  - Incorrect: `<price>100.00<PRICE>`
  - Correct: `<price>100.00<price>`
- **Rule 6**: There must be exactly one root element.

XML APPLICATIONS

SWINBURNE UNIVERSITY OF TECHNOLOGY

XML Applications

- XML files are still simple text files *(just like HTML).*
- When XML is used for a particular project or task, it is called an **"XML application",** such as:
  - XHTML: An XML application of HTML.
  - **KML / GML**: XML applications for geography, e.g., Google Maps)
  - **Ajax**: An XML application for transferring data from server to Web applications.
  - Web Services: An XML application for Service Provision
- XML documents use the file extension **`.xml`**. Specific "XML applications" can use them however they want.

XML Document (continued)

```
<?xml version="1.0"?>
<course>
  <subject>
     <code>COS10005</code>
     <code>COS60002</code>
     <title>Web Development</title>
     <credit>12.5</credit>
  </subject>
  <subject>
     <code>COS20022</code>
     <title>Web Programming</title>
     <credit>12.5</credit>
  </subject>
</course>
```

DOCUMENT TYPE DEFINITION (DTD)

Document Type Definition

- Sometimes XML is too flexible.
- When XML documents are used to exchange data, the format (e.g., structure, elements and attributes) must be fixed.
- Document Type Definition (DTD) is used to specify the allowed format for the data (e.g., structure, elements and attributes).

DTD – Example

```
<!ELEMENT course    (subject+)>
<!ELEMENT subject   (code+,title,credit)>
<!ELEMENT code      (#PCDATA)>
<!ELEMENT title     (#PCDATA)>
<!ELEMENT credit    (#PCDATA)>
```

Content of a <course> element is one or many <subject> elements.

Content of the <code> element is parsed character data.

Content of a <subject> element is one or many <code> elements, a <title> element and a <credit> element.

XML validators follow those rules to validate XML documents.

DTD – Element Declarations

- For each element:

```
<!ELEMENT element_name element_content>
```

- Possible values for `element_content`:
  - `(#PCDATA)`: parsed character data
    - `<!ELEMENT title (#PCDATA)>`
  - `(child)`: one child element type
    - `<!ELEMENT course (subject+)>`
  - `(child1, ..., childn)`: a sequence of child element types
    - `<!ELEMENT subject (code+,title,credit)>`
  - `(child1|...|childn)`: one of the elements

DTD – Element Declarations

```
<!ELEMENT element_name element_content>
```

- For each child element `child`, possible counts can be specified:
  - `child`: exactly one such element
  - `child+`: one or many such elements
  - `child*`: zero or many such elements
  - `child?`: zero or one such element

```
<!ELEMENT subject    (code+,title,credit)>
```

USING JAVASCRIPT TO READ LOCAL XML DATA

SWINBURNE UNIVERSITY OF TECHNOLOGY

XML File

```
<?xml version="1.0" encoding="UTF-8"?>
<Teams>
        <Team>
                <TeamName>Lakers</TeamName>
                <Location>Los Angeles</Location>
                <StarPlayer>Kobe Bryant</StarPlayer>
                <Stadium>Staples Center</Stadium>
        </Team>
        <Team>
                …
        </Team>
        …
</Teams>
```

Step 1: Create A JavaScript Function

```
function parseXML() {
    …
}

//link functions to elements' events
function init() {
    $("#btnExecute").click(parseXML);
}

//the initialise function
$(document).ready(init);
```

Step 2: Create an XML Object

```
function parseXML() {
var xmlhttp;
if (window.XMLHttpRequest) {
  // code for IE7+, Firefox, Chrome, Opera, Safari
  xmlhttp = new XMLHttpRequest();
} else  {
  // code for IE6, IE5
  xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
}
…
```

Step 3a: Setup the Request

…

```
xmlhttp.open(method, url, async);
```

…

where:

`method:` the type of the request, `GET` or `POST`

`url:` the location of the target file

`async:` the request should be handled asynchronously or not, `true` or `false`

Example:

…

```
xmlhttp.open("GET", "nba.xml", false);
```

…

Step 4: Send the Request

…

```
xmlhttp.send();
```

…

This statement will send the request to retrieve the XML data specified before using function `open()`, i.e., `nba.xml`.

Step 5: Retrieve XML Data

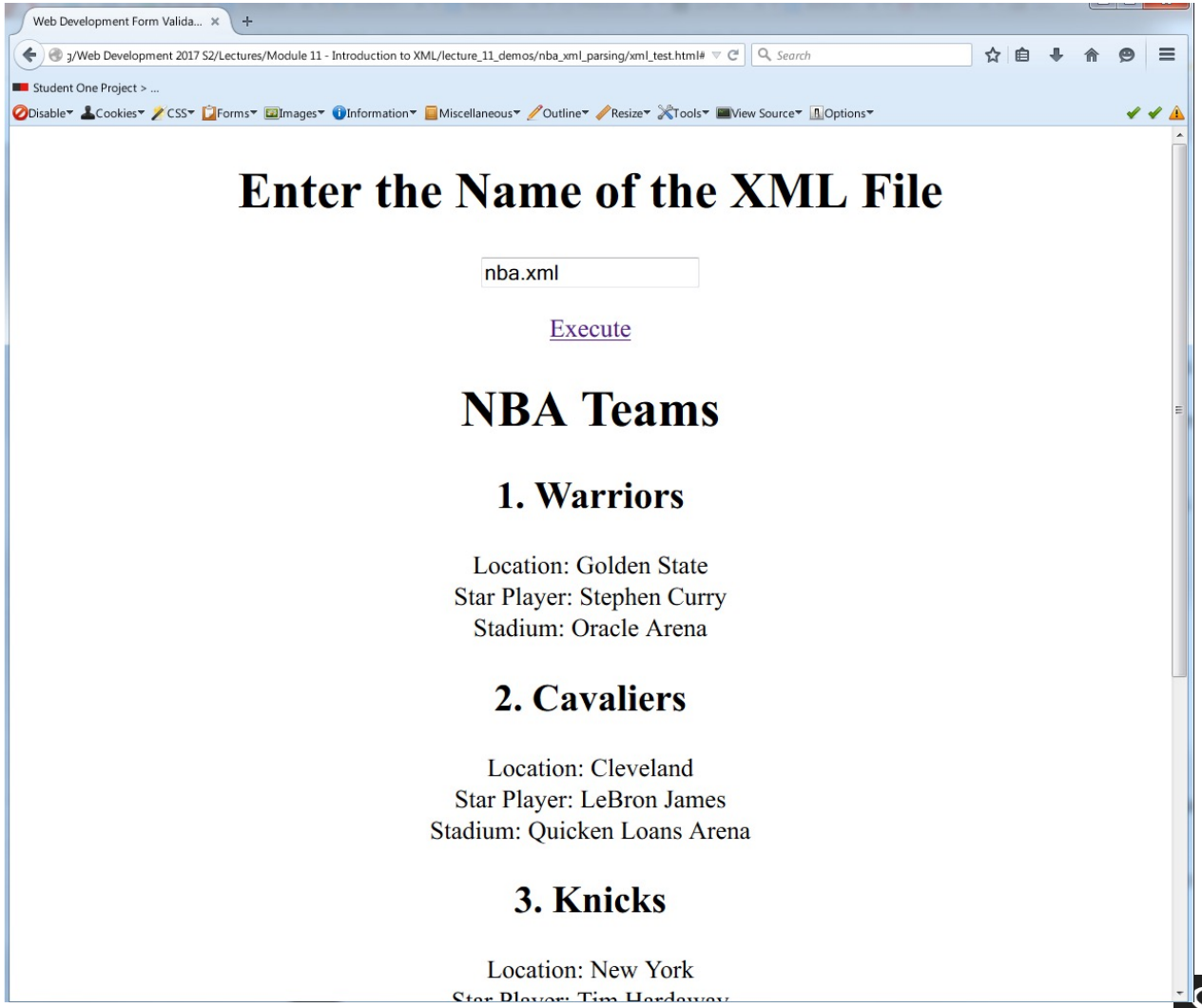
```
…
var xmlDoc = xmlhttp.responseXML;
…
//This statement will retrieve the XML data received
and save it into a variable named xmlDoc.
…
var Teams = xmlDoc.getElementsByTagName("Team");
var TeamNames = xmlDoc.getElementsByTagName("TeamName");
var StarPlayers = xmlDoc.getElementsByTagName("StarPlayer");
var Locations = xmlDoc.getElementsByTagName("Location");
var Stadiums = xmlDoc.getElementsByTagName("Stadium");
…
```

Those statements will retrieve the XML elements using their tag names, i.e., `Team`, `TeamName`, `StarPlayer`, `Location` and `Stadium`.

Step 6: Display XML Data

```
for(var i=0; i<Teams.length; ++i) {
    document.write("<h2>");
    document.write(i+"."+TeamNames[i].childNodes[0].nodeValue);
    document.write("</h2>");

    document.write("Location: ");
    document.write(Locations[i].childNodes[0].nodeValue);
    document.write("<br />");

    document.write("Star Player: ");
    document.write(StarPlayers[i].childNodes[0].nodeValue);
    document.write("<br />");

    document.write("Stadium: ");
    document.write(Stadiums[i].childNodes[0].nodeValue);
    document.write("<br />");
}

//This for loop will display all the retrieved XML data.
```

Result



Web Requests

- Requests made by a client (such as a web browser) to a server using HTTP protocol
- Typically include a URL (Uniform Resource Locator) that points to the location of a resource on the server, and a method that specifies the action to be taken on the resource.
- Most common methods are GET and POST

AJAX(Asynchronous JavaScript and XML)

- It is a technique that allows for asynchronous communication between the client-side (JavaScript) and the server-side (usually a web server).
- AJAX uses a combination of technologies, including JavaScript, XML, and HTML, to dynamically update the content of a web page without requiring a full page refresh (updated asynchronously)
- AJAX allows web pages to update their content without users having to reload the page
- Commonly used in popular web applications like Google Maps, Gmail, Facebook, and Twitter

Comparison of JavaScript & AJAX

Conventional model	AJAX model
The browser sends an HTTP request to the server.	The browser creates a JavaScript call, which then creates a new XMLHttpRequest object.
The web server receives and processes the request.	The new XMLHttpRequest object transfers data between the browser and the web server in an XML format.
The web server sends the requested data to the browser.	The XMLHttpRequest object sends a request for the updated page data to the web server. Subsequently, the latter processes the request and sends it back to the browser.
The browser receives the data from the server and reloads it as an HTML page.	The browser uses JavaScript to process the response and displays the updated content directly on the HTML page without reloading.
Users have to wait until it finishes loading. Therefore, the conventional model increases the load on the server and is more time-consuming.	

AJAX and WEB Requests

- The keystone of AJAX is the XMLHttpRequest object
- XMLHttpRequest object can be used to exchange data with a server
- This means that it is possible to update parts of a web page, without reloading the whole page

AJAX and WEB Requests

- var xhttp = new XMLHttpRequest();
- xhttp.open("GET", "ajax_info.txt", true);
- xhttp.send();

Example of AJAX

```
<!DOCTYPE html>
<html>
<head>
  <title>AJAX Example</title>
  <script>
    function showResult(str) {
      // Create a new XMLHttpRequest object
      var xhttp = new XMLHttpRequest();

      // Set the function to be called when the response is received
      xhttp.onreadystatechange = function() {
        if (this.readyState == 4 && this.status == 200) {
          // Update the webpage with the response text
          document.getElementById("result").innerHTML = this.responseText;
        }
      };

      // Open a new POST request to send the selected item to the server
      xhttp.open("POST", "process.php", true);
      xhttp.setRequestHeader("Content-type", "application/x-www-form-urlenc
      xhttp.send("selected_item=" + str);
    }
  </script>
</head>
<body>
  <h1>Select an item:</h1>
  <select onchange="showResult(this.value)">
    <option value="">-- Select an item --</option>
    <option value="item1">Item 1</option>
    <option value="item2">Item 2</option>
    <option value="item3">Item 3</option>
  </select>
```

This code sends the value of selected drop down list by a user to server.

First onchange event is triggered and the showResult() function is called with the value of the selected item as its argument.

ShowResult () creates a new XMLHttpRequest object and set the onreadystatechange function then we open a new POST request to send the selected item to the server using the open() method and the send() method.

USING AJAX TO READ REMOTE XML DATA

Step 1: Create A JavaScript Function

```
function parseXML() {
    …
}
//link functions to elements' events
function init() {
    $("#btnExecute").click(parseXML);
}
//the initialise function
$(document).ready(init);
```

Same as reading local XML file.

Step 2: Create an XML Object

```
function parseXML() {
var xmlhttp;
if (window.XMLHttpRequest) {
  // code for IE7+, Firefox, Chrome, Opera, Safari
  xmlhttp = new XMLHttpRequest();
} else  {
  // code for IE6, IE5
  xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
}
…
```

Same as reading local XML file.

Step 3: Create An Event Handling Function

```
xmlhttp.onreadystatechange = function() {
      if((xmlhttp.readyState == 4) &&
(xmlhttp.status == 200)) {  //when the xml data is
ready
      //obtain received text
      var xmlDoc=xmlhttp.responseText;
      ////update a specific part of the page
      document.getElementById("pResult").innerHTML
+= xmlDoc;

      document.getElementById("pResult").innerHTML
+= "<br />";
      }
}
```

This function has no name. It is only used to handle the `onreadystatechange` event of the request.

Step 4: Setup the Request

```
…
xmlhttp.open(method, url, async);
```

where:

`method:` the type of the request, `GET` or `POST`

`url:` the location of the target file

`async:` the request should be handled asynchronously or not, `true` or `false`

Example:

```
xmlhttp.open("GET", "xml.php", true);
```

Step 5: Send the Request

```
        …
        xmlhttp.send();
} //end of function parseXML()
```

This statement will send the request to target php page specified before using function `open()`, i.e., `xml.php`.

References

- W3C
  http://www.w3.org/xml/ and http://www.w3.org/TR/xml/
- XML in 10 Points
  https://www.w3.org/XML/1999/XML-in-10-points-19990327
- W3Schools
  (XML Tutorial, Online Tutorial and Reference)
  http://www.w3schools.com/xml/
- xml.com
  http://www.xml.com/

THE END

SWINBURNE UNIVERSITY OF TECHNOLOGY