UNIT III CLIENT SIDE TECHNOLOGIES

- XML
- Document Type Definition
- XML Schema
- Document Object Model
- Presenting XML
- Using XML Parsers: DOM and SAX
- JavaScript Fundamentals
- Evolution of AJAX
- AJAX Framework
- Web applications with AJAX
- AJAX with PHP
- AJAX with Databases

Presented by,
B.Vijayalakshmi
Computer Centre
MIT Campus
Anna University

XML

Xml (e**X**tensible **M**arkup **L**anguage)

- XML is a **markup language** which is **used for storing and transporting data.**
- It was created to **provide an easy to use and store self describing data.**
- It **focuses on data rather than how the data looks.**
- XML became a W3C (W3C stands for World Wide Web Consortium, the main international standards organization for the World Wide Web) recommendation on February 10, 1998.
- XML is **different from HTML**.
- XML is **not a replacement for HTML.**
- XML focuses on data while HTML focuses on how the data looks.
- XML does not depend on software and hardware, it is **platform and programming language independent.**
- Unlike HTML where most of the tags are predefined, **XML doesn't have predefined tags, rather we have to create our own tags.**

Need for XML

- Since there are systems with different-different operating systems having data in different formats. In order to transfer the data between these systems is a difficult task as the data needs to converted in compatible formats before it can be used on other system.
- **With XML, it is so easy to transfer data between** such **systems** as XML doesn't depend on platform and the language.
- XML is a simple document with the data, which can be used to store and transfer data between any systems irrespective of their hardware and software compatibilities.

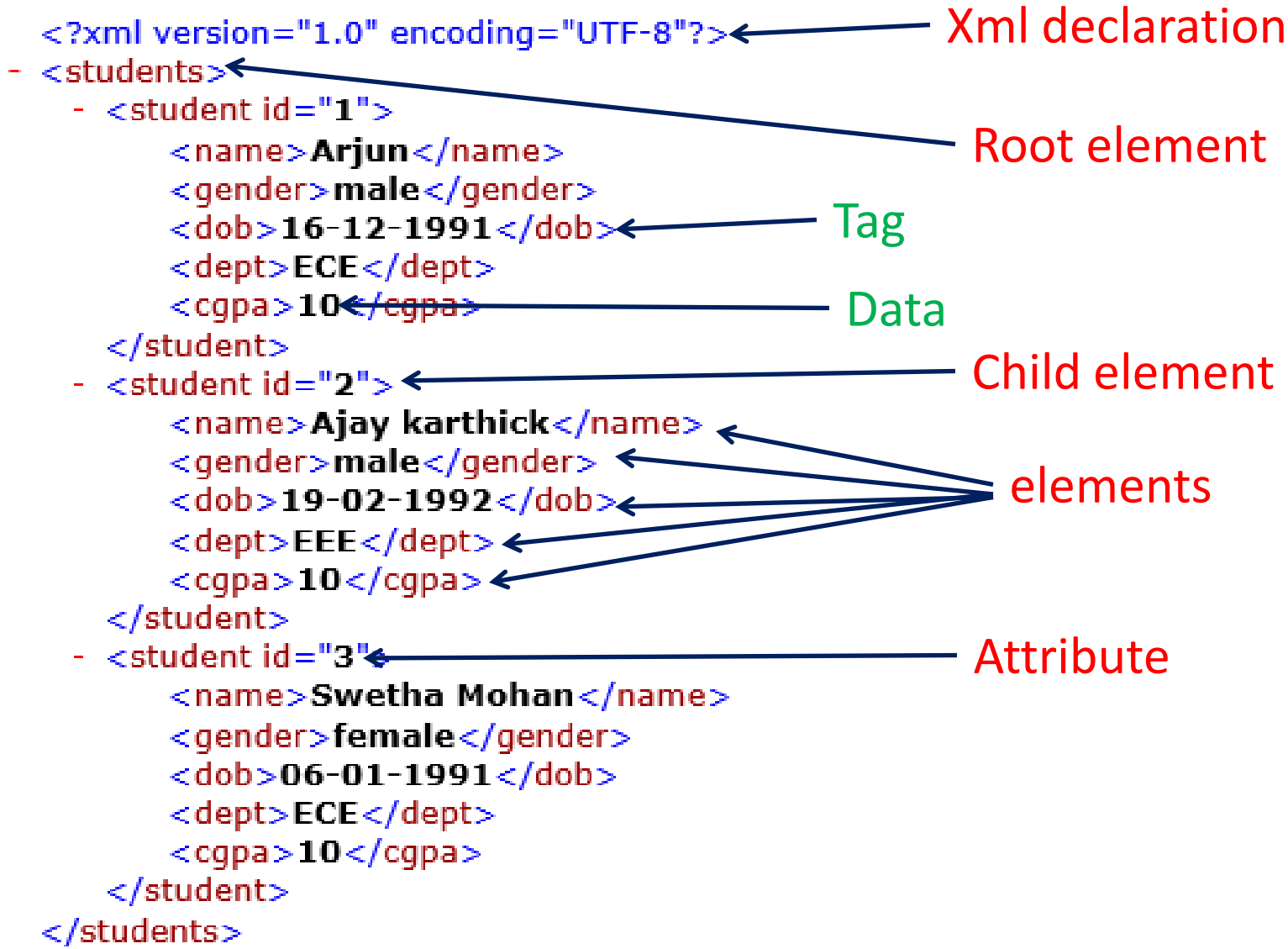
Features and Advantages of XML

- XML separates data from HTML
- XML simplifies data sharing
- XML simplifies data transport
- XML simplifies Platform change
- XML increases data availability
- XML Allows XML Validation
- XML Adapts technology advancements
- XML can be used to create new internet languages
- XML Supports Unicode

Disadvantages of XML

- XML syntax is verbose and redundant compared to other text-based data transmission formats such as **JSON(**JavaScript Object Notation).
- The redundancy in syntax of XML **causes higher storage and transportation cost when the volume of data is large.**
- XML document is less readable compared to other text-based data transmission formats such as JSON.
- **XML doesn’t support array.**
- XML file sizes are usually very large due to its verbose nature, it is totally dependant on who is writing it.

XML - Syntax


<?xml version="1.0" encoding="UTF-8"?>
<students>
<student id="1">
<name>Arjun</name>
<gender>male</gender>
<dob>16-12-1991</dob>
<dept>ECE</dept>
<cgpa>10</cgpa>
</student>
<student id="2">
<name>Ajay karthick</name>
<gender>male</gender>
<dob>19-02-1992</dob>
<dept>EEE</dept>
<cgpa>10</cgpa>
</student>
<student id="3">
<name>Swetha Mohan</name>
<gender>female</gender>
<dob>06-01-1991</dob>
<dept>ECE</dept>
<cgpa>10</cgpa>
</student>
</students>
Xml declaration
Root element
Tag
Data
Child element
elements
Attribute

XML Tree Structure

- XML documents are formed as **element trees**.
- An XML tree starts at a **root element** and branches from the root to **child elements**.
- All elements can have sub elements (child elements):

```
<root>
 <child>
  <subchild>.....</subchild>
 </child>
</root>
```

- The terms parent, child, and sibling are used to describe the relationships between elements.
- Parent have children. Children have parents. Siblings are children on the same level (brothers and sisters).
- All elements can have text content and attributes

XML Declaration

- The XML document can optionally have an XML declaration.

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

  - *version* is the XML version
  - *encoding* specifies the character encoding used in the document.

- ***Syntax Rules for XML Declaration***
  - The XML declaration is case sensitive and must begin with "**<?xml>**" where "**xml**" is written in lower-case.
  - If document contains XML declaration, then it strictly needs to be the first statement of the XML document.
  - An HTTP protocol can override the value of *encoding* that we put in the XML declaration.

XML Tags and Elements

- An XML file is structured by several XML-elements, also called XML-nodes or XML-tags.
- The names of XML-elements are enclosed in triangular brackets < >

Syntax Rules for Tags and Elements

- **Element Syntax** − Each XML-element needs to be closed either with start or with end elements as shown below −
  - <element>....</element> or
  - <element/>
- **Nesting of Elements** − An XML-element can contain multiple XML-elements as its children, but the children elements must not overlap. i.e., an end tag of an element must have the same name as that of the most recent unmatched start tag.

XML Tags and Elements Cont'd

incorrect nested tags

```
<?xml version = "1.0"?>
 <contact-info>
 <company>z-tech
 <contact-info>
</company>
```

correct nested tags

```
<?xml version = "1.0"?>
 <contact-info>
 <company> z-tech
  </company>
<contact-info>
```

XML Tags and Elements Cont'd

Root Element :

- **An XML document can have only one root element.**

For example, following is not a correct XML document, because both the x and y elements occur at the top level without a root element

```
<x>...</x>
<y>...</y>
```

The Following example shows a correctly formed XML document

```
<root>
<x>...</x>
 <y>...</y>
</root>
```

Case Sensitivity :

- The names of XML-elements are case-sensitive.
- For example, **<contact-info>** is different from **<Contact-Info>**

XML Attributes

- An **attribute** specifies a single property for the element, using a name/value pair.
- An XML-element can have one or more attributes. For example Google
- Here **href** is the **attribute name** and **http://www.google.com/** is **attribute value.**

Syntax Rules for XML Attributes

- Attribute names in XML (unlike HTML) are case sensitive.
- That is, *HREF*and *href* are considered two different XML attributes.
- Same attribute cannot have two values in a syntax. The following example shows incorrect syntax because the attribute *b* is specified twice
- ....Attribute names are defined without quotation marks, whereas attribute values must always appear in quotation marks.

XML References

- References usually allow us to add or include additional text or markup in an XML document.
- References always begin with the symbol “&” which is a reserved character and end with the symbol **";".**
- XML has two types of references
  - **Entity References** − An entity reference contains a name between the start and the end delimiters.

    For example **&** where *amp* is *name*.
    The *name* refers to a predefined string of text and/or markup.
  - **Character References** − These contain references, such as **A**, contains a hash mark (“#”) followed by a number. The number always refers to the Unicode code of a character. In this case, 65 refers to alphabet "A".

XML Text

- The names of XML-elements and XML-attributes are case-sensitive
- To avoid character encoding problems, all XML files should be saved as Unicode UTF-8 or UTF-16 files.
- Whitespace characters like blanks, tabs and line-breaks between XML-elements and between the XML-attributes will be ignored.
- Some characters are reserved by the XML syntax itself. Hence, they cannot be used directly.
- To use them, some replacement-entities are used, which are listed below −

Not Allowed Character	Replacement Entity	Character Description
<	<	less than
>	>	greater than
&	&	ampersand
'	'	apostrophe
"	"	quotation mark

XML Comments

- XML comments are just like HTML comments.
- XML Comments add notes or lines for understanding the purpose of an XML code.

Syntax

<!-- Write your comment-->

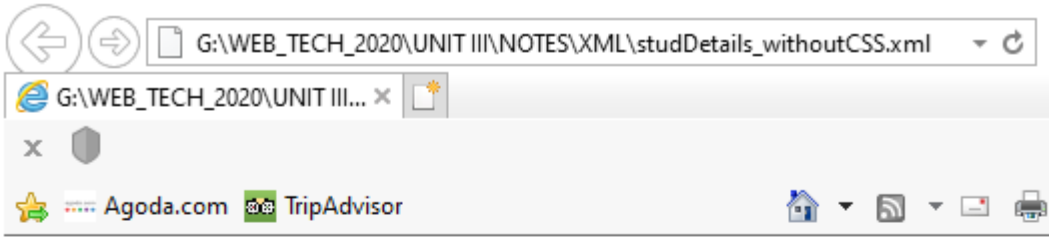
Rules for adding XML comments

- Don't use a comment before an XML declaration.
- We can use a comment anywhere in XML document except within attribute value.
- Don't nest a comment inside the other comment.

Xml file without CSS

Example

```
<?xml version="1.0" encoding="UTF-8"?>
<students>
    <student id="1">
        <name>Arjun</name>
        <gender>male</gender>
        <dob>16-12-1991</dob>
        <dept>ECE</dept>
        <cgpa>10</cgpa>
    </student>
    <student id="2">
        <name>Ajay karthick</name>
        <gender>male</gender>
        <dob>19-02-1992</dob>
        <dept>EEE</dept>
        <cgpa>10</cgpa>
    </student>
    <student id="3">
        <name>Swetha Mohan</name>
        <gender>female</gender>
        <dob>06-01-1991</dob>
        <dept>ECE</dept>
        <cgpa>10</cgpa>
    </student>
</students>
```




```
<?xml version="1.0" encoding="UTF-8"?>
- <students>
  - <student id="1">
        <name>Arjun</name>
        <gender>male</gender>
        <dob>16-12-1991</dob>
        <dept>ECE</dept>
        <cgpa>10</cgpa>
    </student>
  - <student id="2">
        <name>Ajay karthick</name>
        <gender>male</gender>
        <dob>19-02-1992</dob>
        <dept>EEE</dept>
        <cgpa>10</cgpa>
    </student>
  - <student id="3">
        <name>Swetha Mohan</name>
        <gender>female</gender>
        <dob>06-01-1991</dob>
        <dept>ECE</dept>
        <cgpa>10</cgpa>
    </student>
</students>
```

Xml file with CSS

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/css" href="stud.css"?>
<students>
    <student>
        <personal_details>
            <name>Arjun</name>
            <gender>male</gender>
            <dob>16-12-1991</dob>
        </personal_details>
        <academic_details>
            <dept>ECE</dept>
            <cgpa>10</cgpa>
        </academic_details>
    </student>
    <student>
        <personal_details>
            <name>Ajay karthick</name>
            <gender>male</gender>
            <dob>19-02-1992</dob>
        </personal_details>
        <academic_details>
            <dept>EEE</dept>
            <cgpa>10</cgpa>
        </academic_details>
    </student>
```

Example

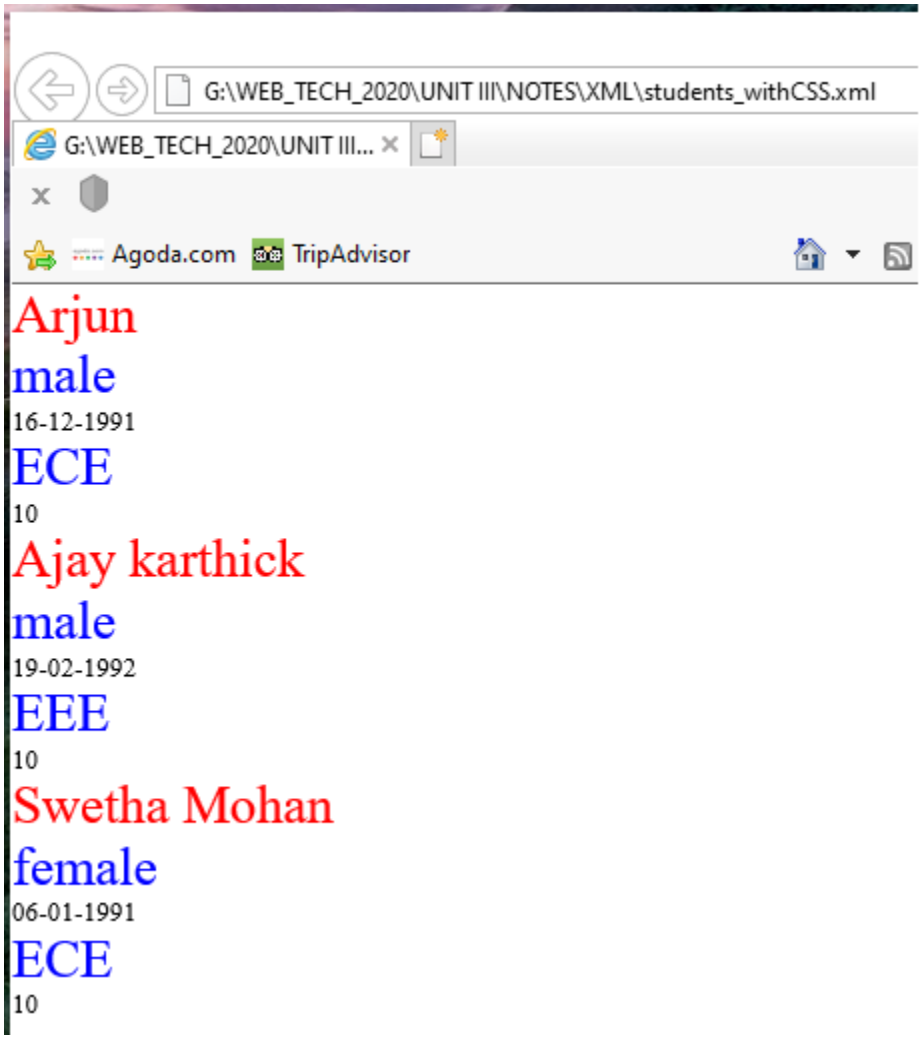
```
<student>
    <personal_details>
        <name>Swetha Mohan</name>
        <gender>female</gender>
        <dob>06-01-1991</dob>
    </personal_details>
    <academic_details>
        <dept>ECE</dept>
        <cgpa>10</cgpa>
    </academic_details>
</student>
</students>
```

Xml file with CSS

stud.css

```
students {
    background-color: #ffffff;
    width: 100%;
}
name{
    display: block;
    color: #ff0000;
    font-size: 20pt;
}
gender{
    display: block;
    color: #0000ff;
    font-size: 20pt;
}
dob {
    display: block;
    color: #000000;
    font-size: 10pt;
}
```

```
dept {
    display: block;
    color: #0000ff;
    font-size: 20pt;
}
cgpa{
    display: block;
    color: #000000;
    font-size: 10pt;
}
```



Document Object Model (DOM)

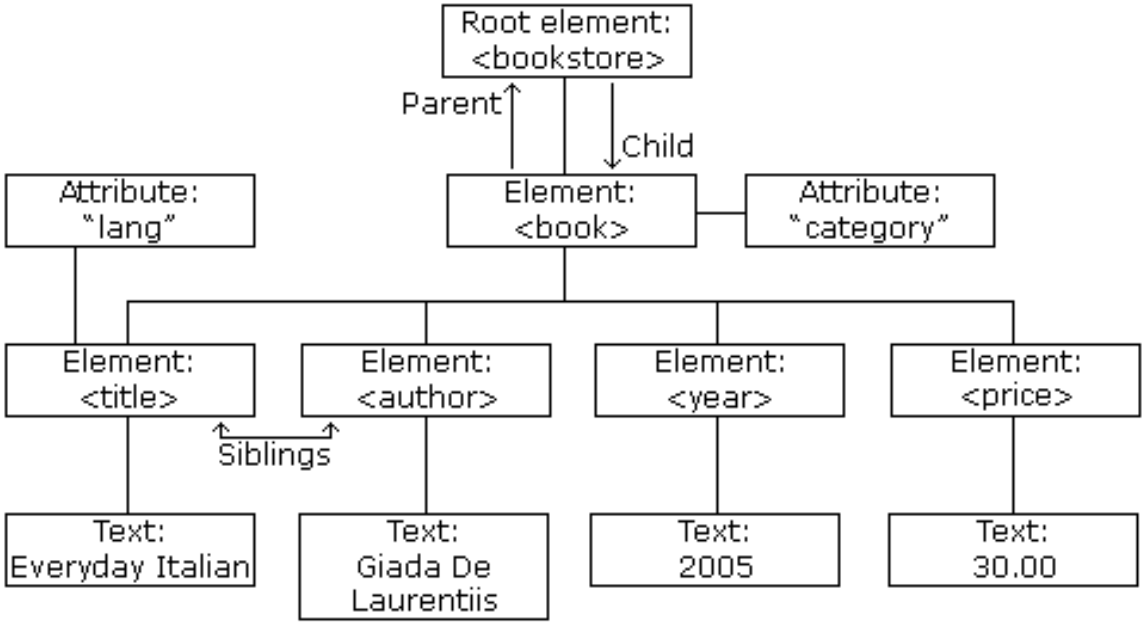
DOM

- DOM defines a standard way to access and manipulate documents.
- DOM is a programming API for HTML and XML documents.
- It defines the logical structure of documents and the way a document is accessed and manipulated.
- As a W3C specification, one important objective for the Document Object Model is to provide a standard programming interface that can be used in a wide variety of environments and applications.
- The Document Object Model can be used with any programming language.

The XML DOM

- The XML DOM (Document Object Model) **defines a standard way for accessing and manipulating XML documents.**
- The DOM **presents an XML document as a tree structure**, with elements, attributes, and text as nodes

DOM Tree



XML DOM Nodes

- According to the XML DOM, everything in an XML document is a **node**:
  - The entire document is a document node
  - Every XML element is an element node
  - The text in the XML elements are text nodes
  - Every attribute is an attribute node
  - Comments are comment nodes

XML DOM Methods

- x.getElementsByTagName(*name*) - get all elements with a specified tag name
- x.appendChild(*node*) - insert a child node to x
- x.removeChild(*node*) - remove a child node from x

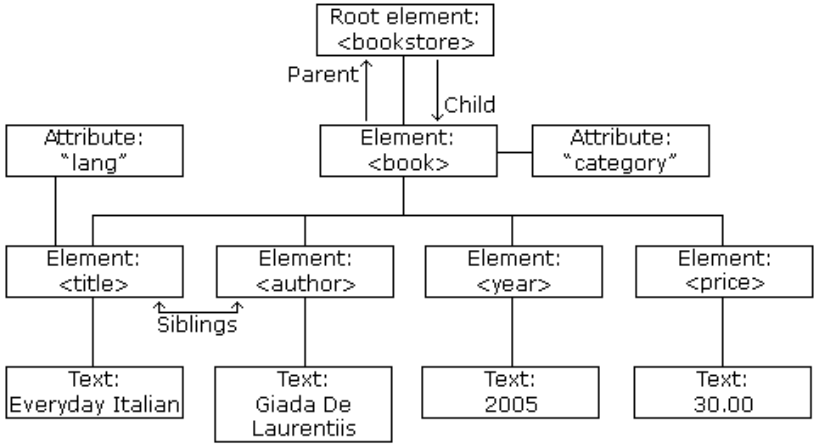
Note: In the list above, x is a node object.

XML DOM Properties

- These are some typical DOM properties:
  - x.nodeName - the name of x
  - x.nodeValue - the value of x
  - x.parentNode - the parent node of x
  - x.childNodes - the child nodes of x
  - x.attributes - the attributes nodes of x

Note: In the list above, x is a node object.

Accessing Nodes



- We can access a node in three ways:
  - By using the getElementsByTagName() method.
  - By looping through (traversing) the nodes tree.
  - By navigating the node tree, using the node relationships

  For Example,

  txt = xmlDoc.getElementsByTagName("title")[0].childNodes[0].nodeValue;
- This code retrieves the text value of the first <title> element in an XML document.

Presenting XML
(xml → html)

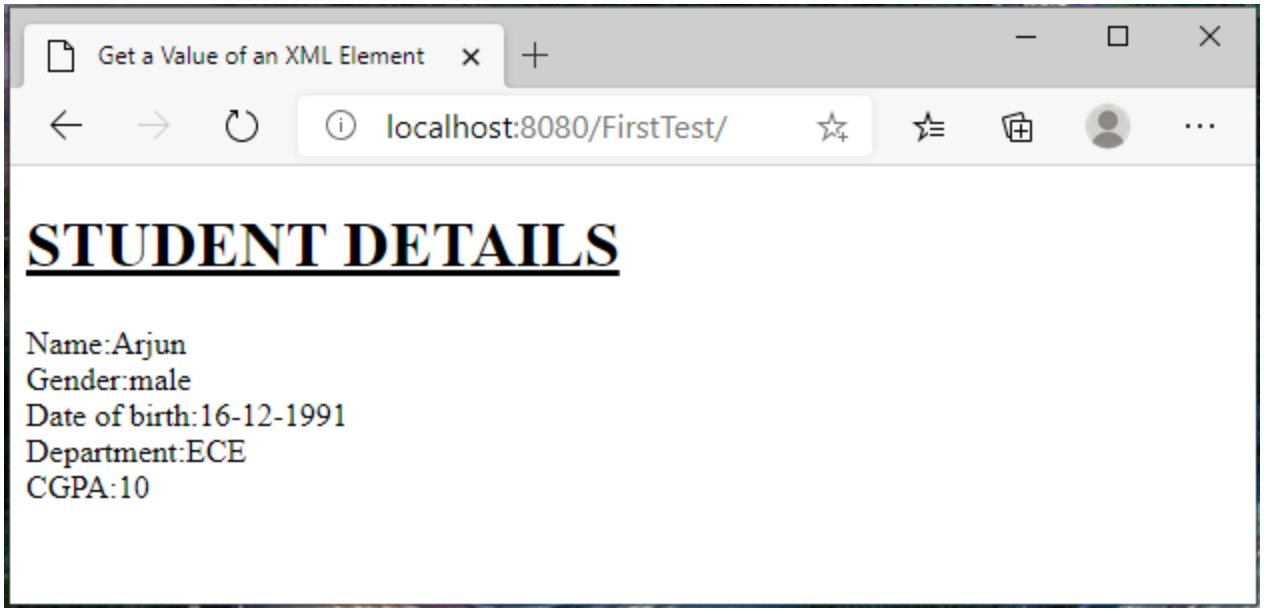
XMLHttpRequest Properties and Methods

Method	Description
new XMLHttpRequest()	Creates a new XMLHttpRequest object
open(method, url, async)	Specifies the type of request method: the type of request: GET or POST url: the file location async: true (asynchronous) or false (synchronous)
send()	Sends a request to the server (used for GET)
send(string)	Sends a request string to the server (used for POST)
onreadystatechange	A function to be called when the readyState property changes
readyState	The status of the XMLHttpRequest 0: request not initialized 1: server connection established 2: request received 3: processing request 4: request finished and response is ready
status	200: OK 404: Page not found
responseText	The response data as a string
responseXML	The response data as XML data

studDetails.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<students>
        <student id="1">
                <name>Arjun</name>
                <gender>male</gender>
                <dob>16-12-1991</dob>
                <dept>ECE</dept>
                <cgpa>10</cgpa>
        </student>
        <student id="2">
                <name>Ajay karthick</name>
                <gender>male</gender>
                <dob>19-02-1992</dob>
                <dept>EEE</dept>
                <cgpa>10</cgpa>
         </student>
        <student id="3">
                <name>Swetha Mohan</name>
                <gender>female</gender>
                <dob>06-01-1991</dob>
                <dept>ECE</dept>
                <cgpa>10</cgpa>
        </student>
</students>
```

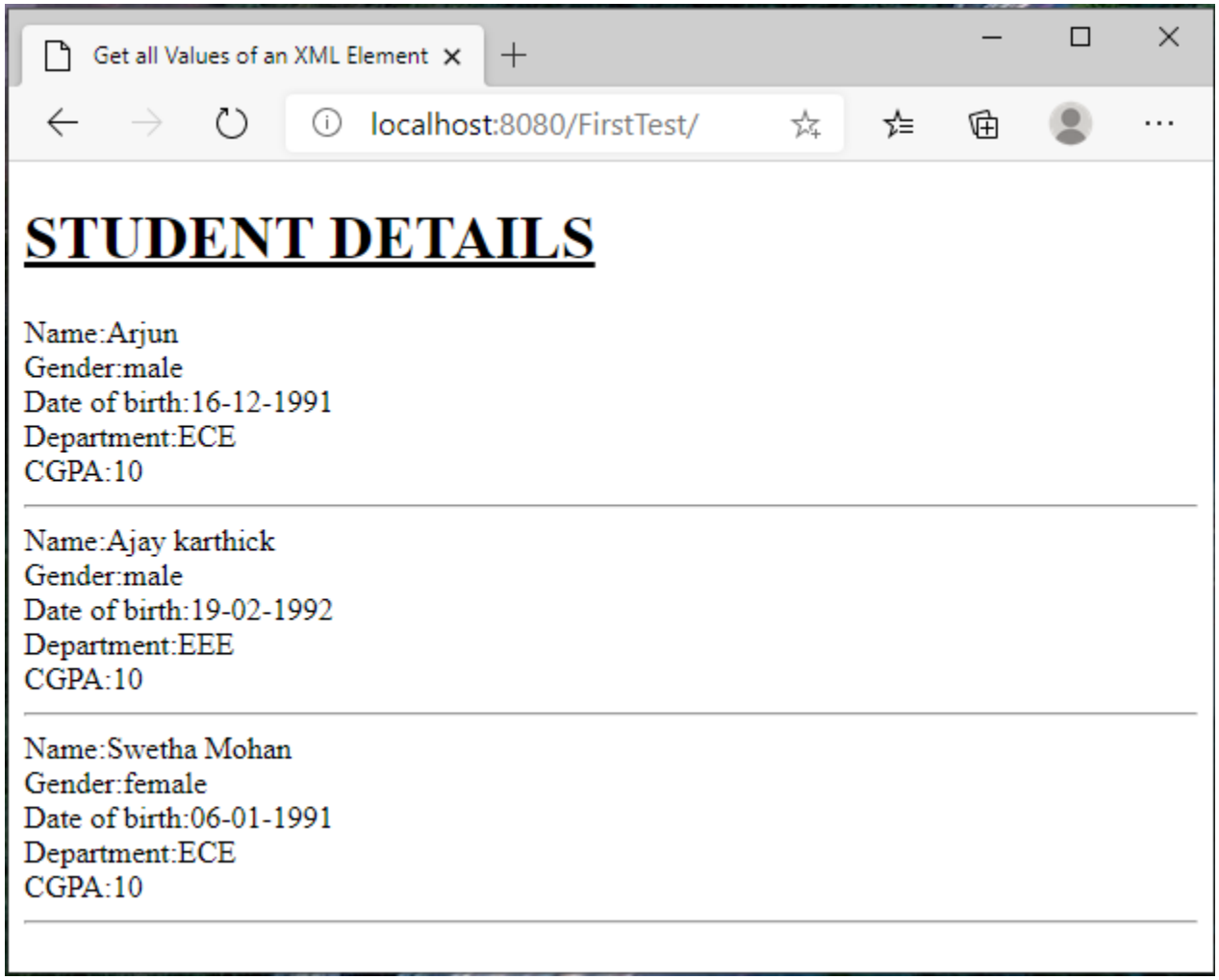
Get a Value of an XML Element



Get a Value of an XML Element

```
<html>
  <head>
    <title>Get a Value of an XML Element</title>
    <script>
        document.write("<h1><U>STUDENT DETAILS</U></H1>");
        var xmlObject,docObject,name,gen,dob,dept,cgpa;
        xmlObject=new XMLHttpRequest();
        xmlObject.open("GET","studDetails.xml",false);
        xmlObject.send();
        docObject=xmlObject.responseXML;
        name=docObject.getElementsByTagName("name")[0].childNodes[0].nodeValue;
            document.write("Name:"+name+"<BR/>");
        gen=docObject.getElementsByTagName("gender")[0].childNodes[0].nodeValue;
            document.write("Gender:"+gen+"<BR/>");
        dob=docObject.getElementsByTagName("dob")[0].childNodes[0].nodeValue;
            document.write("Date of birth:"+dob+"<BR/>");
        dept=docObject.getElementsByTagName("dept")[0].childNodes[0].nodeValue;
            document.write("Department:"+dept+"<BR/>");
        cgpa=docObject.getElementsByTagName("cgpa")[0].childNodes[0].nodeValue;
            document.write("CGPA:"+cgpa+"<BR/>");
    </script>
  </head>
</html>
```

Get all Values of an XML Element



Get all Values of an XML Element

```
<html>
    <head>
        <script>
        var xmlObject,docObject,name,gen,dob,dept,cgpa,ct;
        xmlObject=new XMLHttpRequest();
        xmlObject.open("GET","studDetails.xml",false);
        xmlObject.send();
        docObject=xmlObject.responseXML;
        stuArray=docObject.getElementsByTagName("student");
        for(ct=0;ct<stuArray.length;ct++)
        {
        name=docObject.getElementsByTagName("name")[ct].childNodes[0].nodeValue;
        document.write("Name:"+name+"<BR/>");
        gen=docObject.getElementsByTagName("gender")[ct].childNodes[0].nodeValue;
        document.write("Gender:"+gen+"<BR/>");
        dob=docObject.getElementsByTagName("dob")[ct].childNodes[0].nodeValue;
        document.write("Date of birth:"+dob+"<BR/>");
        dept=docObject.getElementsByTagName("dept")[ct].childNodes[0].nodeValue;
        document.write("Department:"+dept+"<BR/>");
        cgpa=docObject.getElementsByTagName("cgpa")[ct].childNodes[0].nodeValue;
        document.write("CGPA:"+cgpa+"<BR/>");
        document.write("<hr>");
        }
        </script>
    </head>
    <body>
    </body>
</html>
```