Introduction to XML

XML was designed to transport and store data.
HTML was designed to display data.

What is XML?

- XML stands for EXtensible Markup Language
- XML is a markup language much like HTML
- XML was designed to carry data, not to display data
- XML tags are not predefined. You must define your own tags
- XML is designed to be self-descriptive
- XML is a W3C Recommendation

The Difference between XML and HTML

XML is not a replacement for HTML.
XML and HTML were designed with different goals:

- XML was designed to transport and store data, with focus on what data is
- HTML was designed to display data, with focus on how data looks

HTML is about displaying information, while XML is about carrying information.

XML Does Not DO Anything

Maybe it is a little hard to understand, but XML does not DO anything. XML was created to structure, store, and transport information.

The following example is a note to Tove, from Jani, stored as XML:

```
< note>
< to>Tove</to>
< from>Jani</from>
< heading>Reminder</heading>
< body>Don't forget me this weekend!</body>
< /note>
```

The note above is quite self descriptive. It has sender and receiver information, it also has a heading and a message body.
But still, this XML document does not DO anything. It is just information wrapped in tags. Someone must write a piece of software to send, receive or display it.

With XML You Invent Your Own Tags

The tags in the example above (like <to> and <from>) are not defined in any XML standard. These tags are "invented" by the author of the XML document.
That is because the XML language has no predefined tags.
The tags used in HTML are predefined. HTML documents can only use tags defined in the HTML standard (like <p>, <h1>, etc.).
XML allows the author to define his/her own tags and his/her own document structure.

XML is Not a Replacement for HTML
XML is a complement to HTML.

It is important to understand that XML is not a replacement for HTML. In most web applications, XML is used to transport data, while HTML is used to format and display the data.
My best description of XML is this: **XML is a software- and hardware-independent tool for carrying information.**

XML Simplifies Data Sharing
In the real world, computer systems and databases contain data in incompatible formats. XML data is stored in plain text format. This provides a software- and hardware-independent way of storing data. This makes it much easier to create data that can be shared by different applications.

XML Simplifies Data Transport
One of the most time-consuming challenges for developers is to exchange data between incompatible systems over the Internet.
Exchanging data as XML greatly reduces this complexity, since the data can be read by different incompatible applications.

XML Simplifies Platform Changes
Upgrading to new systems (hardware or software platforms), is always time consuming. Large amounts of data must be converted and incompatible data is often lost.
XML data is stored in text format. This makes it easier to expand or upgrade to new operating systems, new applications, or new browsers, without losing data.

XML Tree

XML documents form a tree structure that starts at "the root" and branches to "the leaves".

An Example XML Document: XML documents use a self-describing and simple syntax.

```
< ?xml version="1.0" encoding="UTF-8"?>
< note>
< to>Tove</to>
< from>Jani</from>
< heading>Reminder</heading>
< body>Don't forget me this weekend!</body>
< /note>
```

The first line is the XML declaration. It defines the XML version (1.0).
The next line describes the **root element** of the document (like saying: "this document is a note"):

```
< note>
```

The next 4 lines describe 4 **child elements** of the root (to, from, heading, and body):

```
< to>Tove</to>
< from>Jani</from>
< heading>Reminder</heading>
< body>Don't forget me this weekend!</body>
```

And finally the last line defines the end of the root element:

```
< /note>
```

You can assume, from this example, that the XML document contains a note to Tove from Jani.
Don't you agree that XML is pretty self-descriptive?

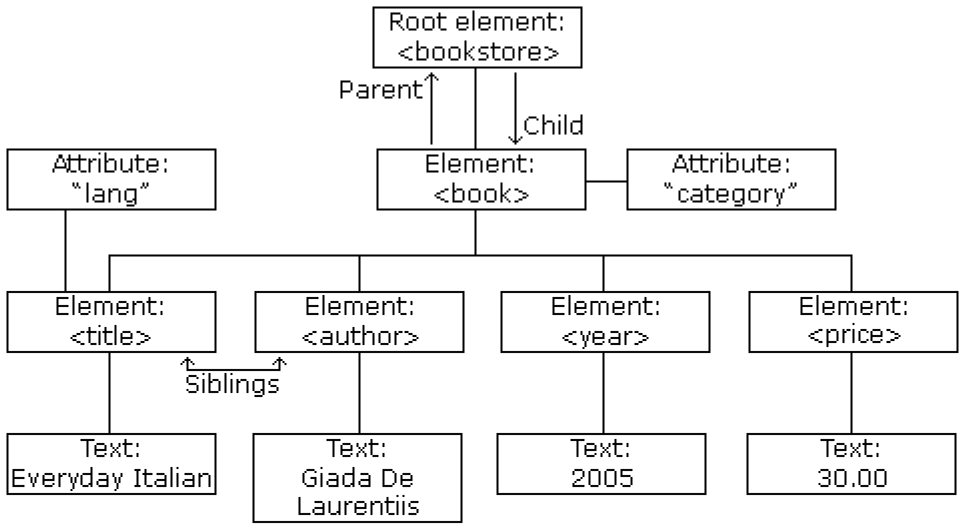
XML Documents Form a Tree Structure

XML documents must contain a **root element**. This element is "the parent" of all other elements.
The elements in an XML document form a document tree. The tree starts at the root and branches to the lowest level of the tree. 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 elements have children. Children on the same level are called siblings (brothers or sisters).
All elements can have text content and attributes (just like in HTML).

Example:



The image above represents one book in the XML below:

```
< bookstore>
< book category="COOKING">
<title lang="en">Everyday Italian</title>
<author>Giada De Laurentiis</author>
<year>2005</year>
<price>30.00</price>
< /book>
< book category="CHILDREN">
<title lang="en">Harry Potter</title>
<author>J K. Rowling</author>
<year>2005</year>
<price>29.99</price>
< /book>
< book category="WEB">
<title lang="en">Learning XML</title>
<author>Erik T. Ray</author>
<year>2003</year>
<price>39.95</price>
< /book>
< /bookstore>
```

The root element in the example is <bookstore>. All <book> elements in the document are contained within <bookstore>.

The <book> element has 4 children: <title>,<author>, <year>, <price>.

XML Syntax Rule

1. All XML Elements Must Have a Closing Tags.
2. XML Tags are Case Sensitive - XML tags are case sensitive. The tag <Letter> is different from the tag <letter>. Opening and closing tags must be written with the same case.
3. All elements **must** be properly nested within each other.
4. XML Documents Must Have a Root Element.
5. XML Attribute Values Must be quoted
   XML elements can have attributes in name/value pairs just like in HTML.
   In XML, the attribute values must always be quoted.
   Study the two XML documents below. The first one is incorrect, the second is correct:

```
< note date=12/11/2007>
< to>Tove</to>
< from>Jani</from>
< /note>
```

```
< note date="12/11/2007">
< to>Tove</to>
< from>Jani</from>
< /note>
```

The error in the first document is that the date attribute in the note element is not quoted.

XML Elements

An XML document contains XML Elements. An XML element is everything from (including) the element's start tag to (including) the element's end tag.

An element can contain:

- other elements
- text
- attributes
- or a mix of all of the above...

```
<bookstore>
<book category="CHILDREN">
<title>Harry Potter</title>
<author>J K. Rowling</author>
<year>2005</year>
<price>29.99</price>
</book>
<book category="WEB">
<title>Learning XML</title>
<author>Erik T. Ray</author>
<year>2003</year>
<price>39.95</price>
</book>
</bookstore>
```

In the example above, <bookstore> and <book> have **element contents**, because they contain other elements. <book> also has an **attribute** (category="CHILDREN"). <title>, <author>, <year>, and <price> have **text content** because they contain text.

Empty XML Elements

An alternative syntax can be used for XML elements with no content. Instead of writing a book element (with no content) like this:

```
<book></book>
```

It can be written like this:

```
<book />
```

This sort of element syntax is called self-closing.

XML Naming Rules

XML elements must follow these naming rules:

- Names can contain letters, numbers, and other characters
- Names cannot start with a number or punctuation character
- Names cannot start with the letters xml (or XML, or Xml, etc)
- Names cannot contain spaces

Any name can be used, no words are reserved.

Best Naming Practices

- Make names descriptive: < first_name>, <last_name>.
- Make names short and simple, like this: <book_title> not like this: < the_title_of_the_book>.
- Avoid "-". If you name something "first-name," some software may think you want to subtract name from first.
- Avoid ".". If you name something "first.name," some software may think that "name" is a property of the object "first."
- Avoid ":". Colons are reserved to be used for something called namespaces (more later).
- Non-English letters like éòá are perfectly legal in XML, but watch out for problems if your software doesn't support them.

XML Elements are Extensible

XML elements can be extended to carry more information. Look at the following XML example:

```
<note>
<to>Tove</to>
<from>Jani</from>
<body>Don't forget me this weekend!</body>
</note>
```

Let's imagine that we created an application that extracted the <to>, <from>, and <body> elements from the XML document to produce this output:

MESSAGE

To: Tove
From: Jani

Don't forget me this weekend!

Imagine that the author of the XML document added some extra information to it:

```
<note>
<date>2008-01-10</date>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

Should the application break or crash?
No. The application should still be able to find the <to>, <from>, and <body> elements in the XML document and produce the same output.
One of the beauties of XML, is that it can be extended without breaking applications.

XML Encoding
XML documents can contain international characters, like Norwegian æøå, or French êèé.
To avoid errors, you schould specify the encoding used, or save your XML files as UTF-8.

Character Encoding

Character encoding defines a unique binary code for each different character used in a document.
In computer terms, character encoding are also called character set, character map, codeset, and code page.

Unicode

Unicode is an industry standard for character encoding of text documents. It defines (nearly) every possible international character by a name and a number.

Unicode has two variants: UTF-8 and UTF-16.
UTF = **U**niversal character set **T**ransformation**F**ormat.
UTF-8 uses a single byte (8-bits) to represent commonly used characters and two (or three) bytes for the rest. UTF-16 uses two bytes (16 bits) for most characters, and three bytes for the rest.

UTF-8 - The Web Standard
UTF-8 is the standard character encoding on the web. UTF-8 is the default character encoding for HTML-5, CSS, JavaScript, PHP, SQL, and XML.

XML Encoding - The first line in an XML document is called the **prolog**:

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

The prolog is optional. Normally it contains the XML version number.
It can also contain information about the encoding used in the document. This prolog specifies UTF-8 encoding:

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

The XML standard states that all XML software must understand both UTF-8 and UTF-16.
UTF-8 is the default for documents without encoding information.
In addition, most XML software systems understand encodings like ISO-8859-1, Windows-1252, and ASCII.

XML Errors
Most often, XML documents are created on one computer, uploaded to a server on a second computer, and displayed by a browser on a third computer. If the encoding is not correctly interpreted by all the three computers, the browser might display meaningless text, or you might get an error message.
Look at these two XML files:Note saved with right encoding andNote saved with wrong encoding.
For high quality XML documents, UTF-8 encoding is be the best to use. UTF-8 covers international characters, and it is also the default, if no encoding is declared.

Conclusion
When you write an XML document:

- Use an XML editor that supports encoding
- Make sure you know what encoding the editor uses
- Describe the encoding in the encoding attribute
- UTF-8 is the safest encoding to use
- UTF-8 is the web standard

XML Document Type
An XML document with correct syntax is called "Well Formed". A "Valid" XML document must also conform to a specified document type.

Well Formed XML Documents
An XML document with correct syntax is "Well Formed". The syntax rules were described in the previous chapters:

- XML documents must have a root element
- XML elements must have a closing tag
- XML tags are case sensitive
- XML elements must be properly nested
- XML attribute values must be quoted

```
< ?xml version="1.0" encoding="UTF-8"?>
< note>
< to>Tove</to>
< from>Jani</from>
< heading>Reminder</heading>
< body>Don't forget me this weekend!</body>
< /note>
```

An XML Validator
To help you check the syntax of your XML files, we have created an XML validator to syntax-check your XML.

Valid XML Documents
A valid XML document is not the same as a well formed XML document.
The first rule, for a valid XML document, is that it must be well formed (see previous paragraph).
The second rule is that a valid XML document must conform to a document type. Rules that defines legal elements and attributes for XML documents are often called document definitions, or document schemas.

When to Use a Document Definition?

A document definition is the easiest way to provide a reference to the legal elements and attributes of a document.
A document definition also provides a common reference that many users (developers) can share.
A document definition provides a standardization that makes life easier.

When NOT to Use a Document Definition?
XML does not require a document definition.
When you are experimenting with XML, or when you are working with small XML files, creating document definitions may be a waste of time.
If you develop applications, wait until the spesification is stable, before you add a document definition. Otherwise your software might stop working, because of validation errors,

Document Definitions
There are different types of document definitions that can be used with XML:

- The original Document Type Definition (DTD)
- The newer, and XML based, XML Schema

XML Parsing

All modern browsers have a built-in XML parser.
An XML parser converts an XML document into an XML DOM object - which can then be manipulated with JavaScript.

Parse an XML String

The following code fragment parses an XML string into an XML DOM object:

```
txt="<bookstore><book>";
txt=txt+"<title>Everyday Italian</title>";
txt=txt+"<author>Giada De Laurentiis</author>";
txt=txt+"<year>2005</year>";
txt=txt+"</book></bookstore>";

if (window.DOMParser)
{
parser=new DOMParser();
xmlDoc=parser.parseFromString(txt,"text/xml");
}
else // Internet Explorer
{
xmlDoc=new ActiveXObject("Microsoft.XMLDOM");
xmlDoc.async=false;
xmlDoc.loadXML(txt);
}
```

Note: Internet Explorer uses the loadXML() method to parse an XML string, while other browsers use the DOMParser object.

Access Across Domains

For security reasons, modern browsers do not allow access across domains.
This means, that both the web page and the XML file it tries to load, must be located on the same server.

DOM
A DOM (Document Object Model) defines a standard way for accessing and manipulating documents.

The XML DOM

The XML DOM defines a standard way for accessing and manipulating XML documents.
The XML DOM views an XML document as a tree-structure.
All elements can be accessed through the DOM tree. Their content (text and attributes) can be modified or deleted, and new elements can be created. The elements, their text, and their attributes are all known as nodes.

The HTML DOM

The HTML DOM defines a standard way for accessing and manipulating HTML documents.
All HTML elements can be accessed through the HTML DOM.

You can learn more about the HTML DOM in our JavaScript tutorial.

Load an XML File - Cross-browser Example

The following example parses an XML document ("note.xml") into an XML DOM object and then extracts some info from it with a JavaScript:

Example

```
< html>
< body>
< h1>W3Schools Internal Note</h1>
< div>
< b>To:</b> <span id="to"></span><br />
< b>From:</b> <span id="from"></span><br />
< b>Message:</b> <span id="message"></span>
< /div>

< script>
if (window.XMLHttpRequest)
{// code for IE7+, Firefox, Chrome, Opera, Safari
xmlhttp=new XMLHttpRequest();
}
else
{// code for IE6, IE5
xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
}
xmlhttp.open("GET","note.xml",false);
xmlhttp.send();
```

```
xmlDoc=xmlhttp.responseXML;

document.getElementById("to").innerHTML=
xmlDoc.getElementsByTagName("to")[0].childNodes[0].nodeValue;
document.getElementById("from").innerHTML=
xmlDoc.getElementsByTagName("from")[0].childNodes[0].nodeValue;
document.getElementById("message").innerHTML=
xmlDoc.getElementsByTagName("body")[0].childNodes[0].nodeValue;
< /script>

< /body>
< /html>
```

Important Note!

To extract the text "Tove" from the <to> element in the XML file above ("note.xml"), the syntax is:

```
getElementsByTagName("to")[0].childNodes[0].nodeValue
```

Notice that even if the XML file contains only ONE <to> element you still have to specify the array index [0]. This is because the getElementsByTagName() method returns an array.

Load an XML String - Cross-browser Example

The following example parses an XML string into an XML DOM object and then extracts some info from it with a JavaScript:

Example

```
< html>
< body>
< h1>W3Schools Internal Note</h1>
< div>
< b>To:</b> <span id="to"></span><br />
< b>From:</b> <span id="from"></span><br />
< b>Message:</b> <span id="message"></span>
< /div>

< script>
txt="<note>";
txt=txt+"<to>Tove</to>";
txt=txt+"<from>Jani</from>";
```

```
txt=txt+"<heading>Reminder</heading>";
txt=txt+"<body>Don't forget me this weekend!</body>";
txt=txt+"</note>";

if (window.DOMParser)
{
parser=new DOMParser();
xmlDoc=parser.parseFromString(txt,"text/xml");
}
else // Internet Explorer
{
xmlDoc=new ActiveXObject("Microsoft.XMLDOM");
xmlDoc.async=false;
xmlDoc.loadXML(txt);
}

document.getElementById("to").innerHTML=
xmlDoc.getElementsByTagName("to")[0].childNodes[0].nodeValue;
document.getElementById("from").innerHTML=
xmlDoc.getElementsByTagName("from")[0].childNodes[0].nodeValue;
document.getElementById("message").innerHTML=
xmlDoc.getElementsByTagName("body")[0].childNodes[0].nodeValue;
< /script>
< /body>
< /html>
```