XML (tutorial from www.3schools.com)

Introduction to XML

XML was designed to describe data, and to focus on what data is.
HTML was designed to display data, and to focus on how data looks.

What you should already know

Before you continue you should have some basic understanding of the following:

- WWW, HTML and the basics of building Web pages
- Web scripting languages like JavaScript or VBScript

If you want to study these subjects first, before you start reading about XML, you can find the tutorials you need at W3Schools' Home Page.

What is XML?

- XML stands for E**X**tensible **M**arkup **L**anguage
- XML is a **markup language** much like HTML
- XML was designed to **describe data**
- XML tags are not predefined in XML. You must **define your own tags**
- XML uses a **Document Type Definition** (DTD) or an **XML Schema** to describe the data
- XML with a DTD or XML Schema is designed to be **self-descriptive**

The main difference between XML and HTML

XML was designed to carry data.

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

XML was designed to describe data and to focus on what data is.
HTML was designed to display data and to focus on how data looks.

HTML is about displaying information, XML is about describing information.

XML does not DO anything

XML was not designed to DO anything.

Maybe it is a little hard to understand, but XML does not DO anything. XML is created to structure, store, and to send 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 has a header, and a message body. It also has sender and receiver information. But still, this XML document does not DO anything. It is just pure information wrapped in XML tags. Someone must write a piece of software to send, receive, or display it.

XML is free and extensible

XML tags are not predefined. You must "invent" your own tags.

The tags used to mark up HTML documents and the structure of HTML documents are predefined. The author of HTML documents can only use tags that are defined in the HTML standard (like <p>, <h1>, etc.).

XML allows the author to define his own tags and his own document structure.

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.

XML is a complement to HTML

XML is not a replacement for HTML.

It is important to understand that XML is not a replacement for HTML. In future Web development it is most likely that XML will be used to describe the data, while HTML will be used to format and display the same data.

My best description of XML is: XML is a cross-platform, software and hardware independent tool for transmitting information.

XML in future Web development

XML is going to be everywhere.

We have been participating in XML development since its creation. It has been amazing to see how quickly the XML standard has been developed, and how quickly a large number of software vendors have adopted the standard.

We strongly believe that XML will be as important to the future of the Web as HTML has been to the foundation of the Web, and that XML will be the most common tool for all data manipulation and data transmission.

XML Joke

Question: When should I use XML?
Answer: When you need a buzzword in your resume.

How can XML be Used?

It is important to understand that XML was designed to store, carry, and exchange data. XML was not designed to display data.

XML can Separate Data from HTML

With XML, your data is stored outside your HTML.

When HTML is used to display data, the data is stored inside your HTML. With XML, data can be stored in separate XML files. This way you can concentrate on using HTML for data layout and display, and be sure that changes in the underlying data will not require any changes to your HTML.

XML data can also be stored inside HTML pages as "Data Islands". You can still concentrate on using HTML only for formatting and displaying the data.

XML is used to Exchange Data

With XML, data can be exchanged between incompatible systems.
In the real world, computer systems and databases contain data in incompatible formats. One of the most time-consuming challenges for developers has been to exchange data between such systems over the Internet.
Converting the data to XML can greatly reduce this complexity and create data that can be read by many different types of applications.

XML and B2B

With XML, financial information can be exchanged over the Internet.

Expect to see a lot about XML and B2B (Business To Business) in the near future.

XML is going to be the main language for exchanging financial information between businesses over the Internet. A lot of interesting B2B applications are under development.

XML can be used to Share Data

With XML, plain text files can be used to share data.

Since XML data is stored in plain text format, XML provides a software- and hardware-independent way of sharing data.
This makes it much easier to create data that different applications can work with. It also makes it easier to expand or upgrade a system to new operating systems, servers, applications, and new browsers.

XML can be used to Store Data

With XML, plain text files can be used to store data.

XML can also be used to store data in files or in databases. Applications can be written to store and retrieve information from the store, and generic applications can be used to display the data.

XML can make your Data more Useful

With XML, your data is available to more users.

Since XML is independent of hardware, software and application, you can make your data available to other than only standard HTML browsers.
Other clients and applications can access your XML files as data sources, like they are accessing databases. Your data can be made available to all kinds of "reading machines" (agents), and it is easier to make your data available for blind people, or people with other disabilities.

XML can be used to Create new Languages

XML is the mother of WAP and WML.
The Wireless Markup Language (WML), used to markup Internet applications for handheld devices like mobile phones, is written in XML.
You can read more about WML in our WML tutorial.

If Developers have Sense

If they DO have sense, all future applications will exchange their data in XML.

The future might give us word processors, spreadsheet applications and databases that can read each other's data in a pure text format, without any conversion utilities in between.
We can only pray that Microsoft and all the other software vendors will agree.

XML Syntax

The syntax rules of XML are very simple and very strict. The rules are very easy to learn, and very easy to use.
Because of this, creating software that can read and manipulate XML is very easy to do.

An example XML document

XML documents use a self-describing and simple syntax.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

The first line in the document - the XML declaration - defines the XML version and the character encoding used in the document. The next line describes the root element of the document (like it was 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>
```

Can you detect from this example that the XML document contains a Note to Tove from Jani? Don't you agree that XML is pretty self-descriptive?

All XML elements must have a closing tag

With XML, it is illegal to omit the closing tag.

In HTML some elements do not have to have a closing tag. The following code is legal in HTML:

```
<p>This is a paragraph
<p>This is another paragraph
```

In XML all elements must have a closing tag, like this:

```
<p>This is a paragraph</p>
<p>This is another paragraph</p>
```

Note: You might have noticed from the previous example that the XML declaration did not have a closing tag. This is not an error. The declaration is not a part of the XML document itself. It is not an XML element, and it should not have a closing tag.

XML tags are case sensitive

Unlike HTML, XML tags are case sensitive.
With XML, the tag <Letter> is different from the tag <letter>.
Opening and closing tags must therefore be written with the same case:

```
<Message>This is incorrect</message>

<message>This is correct</message>
```

All XML elements must be properly nested

Improper nesting of tags makes no sense to XML.
In HTML some elements can be improperly nested within each other like this:

```
<b><i>This text is bold and italic</b></i>
```

In XML all elements must be properly nested within each other like this:

```
<b><i>This text is bold and italic</i></b>
```

All XML documents must have a root tag

The first tag in an XML document is the root tag.
All XML documents must contain a single tag pair to define the root element. All other elements must be nested within the root element.
All elements can have sub elements (children). Sub elements must be correctly nested within their parent element:

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

Attribute values must always be quoted

With XML, it is illegal to omit quotation marks around attribute values.
XML elements can have attributes in name/value pairs just like in HTML. In XML the attribute value must always be quoted. Study the two XML documents below. The first one is incorrect, the second is correct:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<note date=12/11/99>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<note date="12/11/99">
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

The error in the first document is that the date attribute in the note element is not quoted.
This is correct: date="12/11/99". This is incorrect: date=12/11/99.

With XML, White Space is Preserved

With XML, the white space in your document is not truncated.

This is unlike HTML. With HTML, a sentence like this: Hello   my name is Tove,

will be displayed like this: Hello my name is Tove,

because HTML strips off the white space.

With XML, CR / LF is Converted to LF

With XML, a new line is always stored as LF.

Do you know what a typewriter is?. Well, a typewriter is a type of mechanical device they used in the previous century :-)

After you have typed one line of text on a typewriter, you have to manually return the printing carriage to the left margin position and manually feed the paper up one line.

In Windows applications, a new line in the text is normally stored as a pair of CR LF (carriage return, line feed) characters. In Unix applications, a new line is normally stored as a LF character. Some applications use only a CR character to store a new line.

Comments in XML

The syntax for writing comments in XML is similar to that of HTML.
<!-- This is a comment -->

There is Nothing Special about XML

There is nothing special about XML. It is just plain text with the addition of some XML tags enclosed in angle brackets.

Software that can handle plain text can also handle XML. In a simple text editor, the XML tags will be visible and will not be handled specially.

In an XML-aware application however, the XML tags can be handled specially. The tags may or may not be visible, or have a functional meaning, depending on the nature of the application.

XML Elements

XML Elements are extensible and they have relationships.
XML Elements have simple naming rules.

XML Elements are Extensible

XML documents can be extended to carry more information.
Look at the following XML NOTE 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>1999-08-01</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.
XML documents are Extensible.

XML Elements have Relationships

Elements are related as parents and children.

To understand XML terminology, you have to know how relationships between XML elements are named, and how element content is described.

Imagine that this is a description of a book:

Book Title: My First XML
Chapter 1: Introduction to XML
- What is HTML
- What is XML

Chapter 2: XML Syntax
- Elements must have a closing tag
- Elements must be properly nested

Imagine that this XML document describes the book:

```
<book>
<title>My First XML</title>
<prod id="33-657" media="paper"></prod>
<chapter>Introduction to XML
<para>What is HTML</para>
<para>What is XML</para>
</chapter>

<chapter>XML Syntax
<para>Elements must have a closing tag</para>
<para>Elements must be properly nested</para>
</chapter>

</book>
```

Book is the **root element**. Title, prod, and chapter are **child elements** of book. Book is the **parent element** of title, prod, and chapter. Title, prod, and chapter are **siblings** (or **sister elements**) because they have the same parent.

Elements have Content

Elements can have different content types.
An XML element is everything from (including) the element's start tag to (including) the element's end tag.
An element can have element content, mixed content, simple content, or empty content. An element can also have attributes.
In the example above, book has element content, because it contains other elements. Chapter has mixed content because it contains both text and other elements. Para has simple content (or text content) because it contains only text. Prod has empty content, because it carries no information.
In the example above only the prod element has attributes. The attribute named id has the value "33-657". The attribute named media has the value "paper".

Element Naming

XML elements must follow these naming rules:

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

Take care when you "invent" element names and follow these simple rules:
Any name can be used, no words are reserved, but the idea is to make names descriptive. Names with an underscore separator are nice.

Examples: <first_name>, <last_name>.

Avoid "-" and "." in names. It could be a mess if your software tried to subtract name from first (first-name) or think that "name" is a property of the object "first" (first.name).

Element names can be as long as you like, but don't exaggerate. Names should be short and simple, like this: <book_title> not like this: <the_title_of_the_book>.

XML documents often have a corresponding database, in which fields exist corresponding to elements in the XML document. A good practice is to use the naming rules of your database for the elements in the XML documents.

Non-English letters like éòá are perfectly legal in XML element names, but watch out for problems if your software vendor doesn't support them.

The ":" should not be used in element names because it is reserved to be used for something called namespaces (more later).

XML Attributes

XML elements can have attributes in the start tag, just like HTML.
Attributes are used to provide additional information about elements.

XML Attributes

XML elements can have attributes.
From HTML you will remember this: <IMG SRC="computer.gif">. The SRC attribute provides additional information about the IMG element.

In HTML (and in XML) attributes provide additional information about elements:

```
<img src="computer.gif">
<a href="demo.asp">
```

Attributes often provide information that is not a part of the data. In the example below, the file type is irrelevant to the data, but important to the software that wants to manipulate the element:

```
<file type="gif">computer.gif</file>
```

Quote Styles, "female" or 'female'?

Attribute values must always be enclosed in quotes, but either single or double quotes can be used. For a person's sex, the person tag can be written like this:

```
<person sex="female">
```

or like this:

```
<person sex='female'>
```

Double quotes are the most common, but sometimes (if the attribute value itself contains quotes) it is necessary to use single quotes, like in this example:

```
<gangster name='George "Shotgun" Ziegler'>
```

Use of Elements vs. Attributes

Data can be stored in child elements or in attributes.

Take a look at these examples:

```
<person sex="female">
  <firstname>Anna</firstname>
  <lastname>Smith</lastname>
</person>
```

```
<person>
  <sex>female</sex>
  <firstname>Anna</firstname>
  <lastname>Smith</lastname>
</person>
```

In the first example sex is an attribute. In the last, sex is a child element. Both examples provide the same information.
There are no rules about when to use attributes, and when to use child elements. My experience is that attributes are handy in HTML, but in XML you should try to avoid them. Use child elements if the information feels like data.

My Favorite Way

I like to store data in child elements.

The following three XML documents contain exactly the same information:

A date attribute is used in the first example:

```
<note date="12/11/99">
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

A date element is used in the second example:

```
<note>
<date>12/11/99</date>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

An expanded date element is used in the third: (THIS IS MY FAVORITE):

```
<note>
<date>
  <day>12</day>
  <month>11</month>
  <year>99</year>
</date>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

Avoid using attributes?

Should you avoid using attributes?
Here are some of the problems using attributes:

- attributes cannot contain multiple values (child elements can)
- attributes are not easily expandable (for future changes)
- attributes cannot describe structures (child elements can)
- attributes are more difficult to manipulate by program code
- attribute values are not easy to test against a DTD

If you use attributes as containers for data, you end up with documents that are difficult to read and maintain. Try to use **elements** to describe data. Use attributes only to provide information that is not relevant to the data.
Don't end up like this (if you think this looks like XML, you have not understood the point):

```
<note day="12" month="11" year="99"
to="Tove" from="Jani" heading="Reminder"
body="Don't forget me this weekend!">
</note>
```

An Exception to my Attribute rule

Rules always have exceptions.

My rule about attributes has one exception:

Sometimes I assign ID references to elements. These ID references can be used to access XML elements in much the same way as the NAME or ID attributes in HTML. This example demonstrates this:

```
<messages>
  <note id="p501">
    <to>Tove</to>
    <from>Jani</from>
    <heading>Reminder</heading>
    <body>Don't forget me this weekend!</body>
  </note>

  <note id="p502">
    <to>Jani</to>
    <from>Tove</from>
    <heading>Re: Reminder</heading>
    <body>I will not!</body>
  </note>
</messages>
```

The ID in these examples is just a counter, or a unique identifier, to identify the different notes in the XML file, and not a part of the note data.
What I am trying to say here is that metadata (data about data) should be stored as attributes, and that data itself should be stored as elements.

XML Validation

XML with correct syntax is Well Formed XML.
XML validated against a DTD is Valid XML.

"Well Formed" XML documents

A "Well Formed" XML document has correct XML syntax.
A "Well Formed" XML document is a document that conforms to the XML syntax rules that were described in the previous chapters:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

"Valid" XML documents

A "Valid" XML document also conforms to a DTD.
A "Valid" XML document is a "Well Formed" XML document, which also conforms to the rules of a Document Type Definition (DTD):

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE note SYSTEM "InternalNote.dtd">
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

XML DTD

A DTD defines the legal elements of an XML document.
The purpose of a DTD is to define the legal building blocks of an XML document. It defines the document structure with a list of legal elements. You can read more about DTD, and how to validate your XML documents in our DTD tutorial.

XML Schema

XSchema is an XML based alternative to DTD.
W3C supports an alternative to DTD called XML Schema. You can read more about XML Schema in our Schema tutorial.

Errors will Stop you

Errors in XML documents will stop the XML program.
The W3C XML specification states that a program should not continue to process an XML document if it finds a validation error. The reason is that XML software should be easy to write, and that all XML documents should be compatible.
With HTML it was possible to create documents with lots of errors (like when you forget an end tag). One of the main reasons that HTML browsers are so big and incompatible, is that they have their own ways to figure out what a document should look like when they encounter an HTML error.
With XML this should not be possible.

Viewing XML Files

To view an XML document in IE 5.0 (and higher) you can click on a link, type the URL in the address bar, or double-click on the name of an XML file in a files folder. If you open an XML document in IE, it will display the document with color coded root and child elements. A plus (+) or minus sign (-) to the left of the elements can be clicked to expand or collapse the element structure. If you want to view the raw XML source, you must select "View Source" from the browser menu.
To view an XML document in Netscape 6 you'll have to open the XML file and then right-click in XML file and select "View Page Source". If you open an XML document in Netscape 6, it will display the document with color coded root and child elements.
Look at this XML file: note.xml
Note: Do not expect XML files to be formatted like an HTML document !

Viewing an invalid XML file

If an erroneous XML file is opened, the browser will report the error.
Look at this XML file: note_error.xml

Why does XML display like this?

XML documents do not carry information about how to display the data.
Since XML tags are "invented" by the author of the XML document, browsers do not know if a tag like <table> describes an HTML table or a dining table.
Without any information about how to display the data, most browsers will just display the XML document as it is.
In the next chapters, we will take a look at different solutions to the display problem, using CSS, XSL, JavaScript, and XML Data Islands.

Displaying XML with CSS

With CSS (Cascading Style Sheets) you can add display information to an XML document.

Displaying your future XML files with CSS?

Will you be using CSS to format your future XML files?
No, we don't think so! But we could not resist giving it a try:
Take a look at this pure XML file: The CD Catalog
Then look at this style sheet: The CSS file
Finally, view: The CD Catalog formatted with the CSS file
Here is a fraction of the XML file, with an added CSS stylesheet reference:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/css" href="cd_catalog.css"?>
<CATALOG>
  <CD>
    <TITLE>Empire Burlesque</TITLE>
    <ARTIST>Bob Dylan</ARTIST>
    <COUNTRY>USA</COUNTRY>
    <COMPANY>Columbia</COMPANY>
    <PRICE>10.90</PRICE>
    <YEAR>1985</YEAR>
  </CD>
  <CD>
    <TITLE>Hide your heart</TITLE>
    <ARTIST>Bonnie Tyler</ARTIST>
    <COUNTRY>UK</COUNTRY>
    <COMPANY>CBS Records</COMPANY>
    <PRICE>9.90</PRICE>
    <YEAR>1988</YEAR>
  </CD>
.
.
.
.
</CATALOG>
```

We DO NOT believe that formatting XML with CSS is the future of the Web. Even if it looks right to use CSS this way, we DO believe that formatting with XSL will be the new standard (as soon as the main browsers support it).

Creating your future Homepages with XML?

Will you be writing your future Homepages in XML?
No, we don't think you will! But we could not resist giving it a try : A homepage written in XML.
We DO NOT believe that XML will be used to create future Homepages.
We DO believe however, that XHTML - HTML defined as XML will do the trick: Please, read our XHTML tutorial.

Displaying XML with XSL

With XSL you can add display information to your XML document.

Displaying XML with XSL

XSL is the preferred style sheet language of XML.
XSL (the eXtensible Stylesheet Language) is far more sophisticated than CSS. One way to use XSL is to transform XML into HTML before it is displayed by the browser as demonstrated in these examples:
If you have Netscape 6 or IE 5 or higher you can view the XML file and the XSL style sheet.
View the result in IE 6
View the result in IE 5

A shortened copy of the file is shown below. Note the XSL reference in line two:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href="simple.xsl"?>
<breakfast_menu>
  <food>
    <name>Belgian Waffles</name>
    <price>$5.95</price>
    <description>
       two of our famous Belgian Waffles
    </description>
    <calories>650</calories>
  </food>
</breakfast_menu>
```

If you want to learn more about XSL, please visit our XSL tutorial.

XML in Data Islands

With Internet Explorer 5.0 and higher, XML can be embedded within HTML pages in Data Islands.

XML Embedded in HTML

The unofficial <xml> tag is used to embed XML data within HTML.

XML data can be embedded directly in an HTML page like this:

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

Or a separate XML file can be embedded like this:

```
<xml id="note" src="note.xml">
</xml>
```

Note that the <xml> tag is an HTML element, not an XML element.

XML Namespaces

XML Namespaces provides a method to avoid element name conflicts.

Name Conflicts

Since element names in XML are not fixed, very often a name conflict will occur when two different documents use the same names describing two different types of elements.
This XML document carries information in a table:

```
<table>
   <tr>
   <td>Apples</td>
   <td>Bananas</td>
   </tr>
</table>
```

This XML document carries information about a table (a piece of furniture):

```
<table>
   <name>African Coffee Table</name>
   <width>80</width>
   <length>120</length>
</table>
```

If these two XML documents were added together, there would be an element name conflict because both documents contain a <table> element with different content and definition.

Solving Name Conflicts using a Prefix

This XML document carries information in a table:

```
<h:table>
   <h:tr>
   <h:td>Apples</h:td>
   <h:td>Bananas</h:td>
   </h:tr>
</h:table>
```

This XML document carries information about a piece of furniture:

```
<f:table>
   <f:name>African Coffee Table</f:name>
   <f:width>80</f:width>
   <f:length>120</f:length>
</f:table>
```

Now the element name conflict is gone because the two documents use a different name for their <table> element (<h:table> and <f:table>).
By using a prefix, we have created two different types of <table> elements.

Using Namespaces

This XML document carries information in a table:

```
<h:table xmlns:h="http://www.w3.org/TR/html4/">
   <h:tr>
   <h:td>Apples</h:td>
   <h:td>Bananas</h:td>
   </h:tr>
</h:table>
```

This XML document carries information about a piece of furniture:

```
<f:table xmlns:f="http://www.w3schools.com/furniture">
   <f:name>African Coffee Table</f:name>
   <f:width>80</f:width>
   <f:length>120</f:length>
</f:table>
```

Instead of using only prefixes, an **xmlns attribute** has been added to the <table> tag to give the element prefix a **qualified name** associated with a **namespace**.

The Namespace Attribute

The namespace attribute is placed in the start tag of an element and has the following syntax:

```
xmlns:namespace-prefix="namespace"
```

In the examples above, the namespace itself is defined using an Internet address:

```
xmlns:f="http://www.w3schools.com/furniture"
```

The W3C namespace specification states that the namespace itself should be a **Uniform Resource Identifier (URI)**.
When a namespace is defined in the start tag of an element, all child elements with the same prefix are associated with the same namespace.
Note that the address used to identify the namespace, is not used by the parser to look up information. The only purpose is to give the namespace a unique name. However, very often companies use the namespace as a pointer to a real Web page containing information about the namespace. Try to go to http://www.w3.org/TR/html4/.

Uniform Resource Identifiers

A **Uniform Resource Identifier** (URI) is a string of characters which identifies an Internet Resource. The most common URI is the **Uniform Resource Locator** (URL) which identifies an Internet domain address. Another, not so common type of URI is the **Universal Resource Name** (URN). In our examples we will only use URLs.
Since our furniture example uses an internet address to identify its namespace, we can be sure that our namespace is unique.

Default Namespaces

Defining a default namespace for an element saves us from using prefixes in all the child elements. It has the following syntax:

```
<element xmlns="namespace">
```

This XML document carries information in a table:

```
<table xmlns="http://www.w3.org/TR/html4/">
   <tr>
   <td>Apples</td>
   <td>Bananas</td>
   </tr>
</table>
```

This XML document carries information about a piece of furniture:

```
<table xmlns="http://www.w3schools.com/furniture">
   <name>African Coffee Table</name>
   <width>80</width>
   <length>120</length>
</table>
```