http://www.w3schools.com/php/php_ajax_xml.asp

PHP Example - AJAX and XML

AJAX can be used for interactive communication with an XML file.

AJAX XML Example

The following example will demonstrate how a web page can fetch information from an XML file with AJAX:

Example

CD info will be listed here...

Example Explained - The HTML Page

When a user selects a CD in the dropdown list above, a function called "showCD()" is executed. The function is triggered by the "onchange" event:

```
<html>
<head>
<script>
function showCD(str) {
  if (str=="") {
    document.getElementById("txtHint").innerHTML="";
    return;
  }
  if (window.XMLHttpRequest) {
    // code for IE7+, Firefox, Chrome, Opera, Safari
    xmlhttp=new XMLHttpRequest();
  } else {  // code for IE6, IE5
    xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
  }
  xmlhttp.onreadystatechange=function() {
    if (xmlhttp.readyState==4 && xmlhttp.status==200) {
      document.getElementById("txtHint").innerHTML=xmlhttp.responseText;
    }
  }
  xmlhttp.open("GET","getcd.php?q="+str,true);
  xmlhttp.send();
}
```

```
</script>
</head>
<body>

<form>
Select a CD:
<select name="cds" onchange="showCD(this.value)">
<option value="">Select a CD:</option>
<option value="Bob Dylan">Bob Dylan</option>
<option value="Bonnie Tyler">Bonnie Tyler</option>
<option value="Dolly Parton">Dolly Parton</option>
</select>
</form>
<div id="txtHint"><b>CD info will be listed here...</b></div>

</body>
</html>
```

The showCD() function does the following:

- Check if a CD is selected
- Create an XMLHttpRequest object
- Create the function to be executed when the server response is ready
- Send the request off to a file on the server
- Notice that a parameter (q) is added to the URL (with the content of the dropdown list)

The PHP File

The page on the server called by the JavaScript above is a PHP file called "getcd.php".

The PHP script loads an XML document, "cd_catalog.xml", runs a query against the XML file, and returns the result as HTML:

```
<?php
$q=$_GET["q"];

$xmlDoc = new DOMDocument();
$xmlDoc->load("cd_catalog.xml");

$x=$xmlDoc->getElementsByTagName('ARTIST');

for ($i=0; $i<=$x->length-1; $i++) {
  //Process only element nodes
  if ($x->item($i)->nodeType==1) {
    if ($x->item($i)->childNodes->item(0)->nodeValue == $q) {
      $y=($x->item($i)->parentNode);
    }
  }
}

$cd=($y->childNodes);

for ($i=0;$i<$cd->length;$i++) {
  //Process only element nodes
  if ($cd->item($i)->nodeType==1) {
    echo("<b>" . $cd->item($i)->nodeName . ":</b> ");
    echo($cd->item($i)->childNodes->item(0)->nodeValue);
    echo("<br>");
  }
}
?>
```

When the CD query is sent from the JavaScript to the PHP page, the following happens:

1. PHP creates an XML DOM object
2. Find all <artist> elements that matches the name sent from the JavaScript
3. Output the album information (send to the "txtHint" placeholder)

http://www.w3schools.com/ajax/ajax_xmlfile.asp

AJAX XML Example

AJAX can be used for interactive communication with an XML file.

AJAX XML Example

The following example will demonstrate how a web page can fetch information from an XML file with AJAX:

Example

Title	Artist
Empire Burlesque	Bob Dylan
Hide your heart	Bonnie Tyler
Greatest Hits	Dolly Parton
Still got the blues	Gary Moore
Eros	Eros Ramazzotti
One night only	Bee Gees
Sylvias Mother	Dr.Hook
Maggie May	Rod Stewart
Romanza	Andrea Bocelli
When a man loves a woman	Percy Sledge
Black angel	Savage Rose
1999 Grammy Nominees	Many
For the good times	Kenny Rogers
Big Willie style	Will Smith

Tupelo Honey	Van Morrison
Soulsville	Jorn Hoel
The very best of	Cat Stevens
Stop	Sam Brown
Bridge of Spies	T'Pau
Private Dancer	Tina Turner
Midt om natten	Kim Larsen
Pavarotti Gala Concert	Luciano Pavarotti
The dock of the bay	Otis Redding
Picture book	Simply Red
Red	The Communards
Unchain my heart	Joe Cocker

Try it yourself »

Example Explained - The stateChange() Function

When a user clicks on the "Get CD info" button above, the loadXMLDoc() function is executed.

The loadXMLDoc() function creates an XMLHttpRequest object, adds the function to be executed when the server response is ready, and sends the request off to the server.

When the server response is ready, an HTML table is built, nodes (elements) are extracted from the XML file, and it finally updates the txtCDInfo placeholder with the HTML table filled with XML data:

```
<!DOCTYPE html>
<html>
<head>
<script>
function loadXMLDoc(url)
{
var xmlhttp;
var txt,x,xx,i;
if (window.XMLHttpRequest)
  {// code for IE7+, Firefox, Chrome, Opera, Safari
  xmlhttp=new XMLHttpRequest();
  }
else
  {// code for IE6, IE5
  xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
  }
xmlhttp.onreadystatechange=function()
  {
  if (xmlhttp.readyState==4 && xmlhttp.status==200)
    {
    txt="<table border='1'><tr><th>Title</th><th>Artist</th></tr>";
    x=xmlhttp.responseXML.documentElement.getElementsByTagName("CD");
    for (i=0;i<x.length;i++)
      {
      txt=txt + "<tr>";
      xx=x[i].getElementsByTagName("TITLE");
        {
        try
          {
          txt=txt + "<td>" + xx[0].firstChild.nodeValue + "</td>";
          }
        catch (er)
          {
          txt=txt + "<td> </td>";
          }
        }
      xx=x[i].getElementsByTagName("ARTIST");
        {
        try
          {
          txt=txt + "<td>" + xx[0].firstChild.nodeValue + "</td>";
          }
        catch (er)
          {
          txt=txt + "<td> </td>";
          }
        }
      txt=txt + "</tr>";
      }
    txt=txt + "</table>";
    document.getElementById('txtCDInfo').innerHTML=txt;
    }
  }
xmlhttp.open("GET",url,true);
xmlhttp.send();
}
</script>
</head>
<body>

<div id="txtCDInfo">
<button onclick="loadXMLDoc('cd_catalog.xml')">Get CD info</button>
</div>

</body>
</html>
```

The AJAX Server Page

The page on the server used in the example above, is an XML file called "cd_catalog.xml":

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- Edited by XMLSpy□ -->
<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>
	<CD>
		<TITLE>Greatest Hits</TITLE>
		<ARTIST>Dolly Parton</ARTIST>
		<COUNTRY>USA</COUNTRY>
```

```xml
		<COMPANY>RCA</COMPANY>
		<PRICE>9.90</PRICE>
		<YEAR>1982</YEAR>
	</CD>
	<CD>
		<TITLE>Still got the blues</TITLE>
		<ARTIST>Gary Moore</ARTIST>
		<COUNTRY>UK</COUNTRY>
		<COMPANY>Virgin records</COMPANY>
		<PRICE>10.20</PRICE>
		<YEAR>1990</YEAR>
	</CD>
	<CD>
		<TITLE>Eros</TITLE>
		<ARTIST>Eros Ramazzotti</ARTIST>
		<COUNTRY>EU</COUNTRY>
		<COMPANY>BMG</COMPANY>
		<PRICE>9.90</PRICE>
		<YEAR>1997</YEAR>
	</CD>
	<CD>
		<TITLE>One night only</TITLE>
		<ARTIST>Bee Gees</ARTIST>
		<COUNTRY>UK</COUNTRY>
		<COMPANY>Polydor</COMPANY>
		<PRICE>10.90</PRICE>
		<YEAR>1998</YEAR>
	</CD>
	<CD>
		<TITLE>Sylvias Mother</TITLE>
		<ARTIST>Dr.Hook</ARTIST>
		<COUNTRY>UK</COUNTRY>
		<COMPANY>CBS</COMPANY>
		<PRICE>8.10</PRICE>
		<YEAR>1973</YEAR>
	</CD>
	<CD>
		<TITLE>Maggie May</TITLE>
		<ARTIST>Rod Stewart</ARTIST>
		<COUNTRY>UK</COUNTRY>
		<COMPANY>Pickwick</COMPANY>
		<PRICE>8.50</PRICE>
		<YEAR>1990</YEAR>
	</CD>
	<CD>
		<TITLE>Romanza</TITLE>
		<ARTIST>Andrea Bocelli</ARTIST>
		<COUNTRY>EU</COUNTRY>
		<COMPANY>Polydor</COMPANY>
		<PRICE>10.80</PRICE>
		<YEAR>1996</YEAR>
	</CD>
	<CD>
		<TITLE>When a man loves a woman</TITLE>
		<ARTIST>Percy Sledge</ARTIST>
		<COUNTRY>USA</COUNTRY>
```

```
        <COMPANY>Atlantic</COMPANY>
        <PRICE>8.70</PRICE>
        <YEAR>1987</YEAR>
</CD>
<CD>
        <TITLE>Black angel</TITLE>
        <ARTIST>Savage Rose</ARTIST>
        <COUNTRY>EU</COUNTRY>
        <COMPANY>Mega</COMPANY>
        <PRICE>10.90</PRICE>
        <YEAR>1995</YEAR>
</CD>
<CD>
        <TITLE>1999 Grammy Nominees</TITLE>
        <ARTIST>Many</ARTIST>
        <COUNTRY>USA</COUNTRY>
        <COMPANY>Grammy</COMPANY>
        <PRICE>10.20</PRICE>
        <YEAR>1999</YEAR>
</CD>
<CD>
        <TITLE>For the good times</TITLE>
        <ARTIST>Kenny Rogers</ARTIST>
        <COUNTRY>UK</COUNTRY>
        <COMPANY>Mucik Master</COMPANY>
        <PRICE>8.70</PRICE>
        <YEAR>1995</YEAR>
</CD>
<CD>
        <TITLE>Big Willie style</TITLE>
        <ARTIST>Will Smith</ARTIST>
        <COUNTRY>USA</COUNTRY>
        <COMPANY>Columbia</COMPANY>
        <PRICE>9.90</PRICE>
        <YEAR>1997</YEAR>
</CD>
<CD>
        <TITLE>Tupelo Honey</TITLE>
        <ARTIST>Van Morrison</ARTIST>
        <COUNTRY>UK</COUNTRY>
        <COMPANY>Polydor</COMPANY>
        <PRICE>8.20</PRICE>
        <YEAR>1971</YEAR>
</CD>
<CD>
        <TITLE>Soulsville</TITLE>
        <ARTIST>Jorn Hoel</ARTIST>
        <COUNTRY>Norway</COUNTRY>
        <COMPANY>WEA</COMPANY>
        <PRICE>7.90</PRICE>
        <YEAR>1996</YEAR>
</CD>
<CD>
        <TITLE>The very best of</TITLE>
        <ARTIST>Cat Stevens</ARTIST>
        <COUNTRY>UK</COUNTRY>
```

```xml
        <COMPANY>Island</COMPANY>
        <PRICE>8.90</PRICE>
        <YEAR>1990</YEAR>
    </CD>
    <CD>
        <TITLE>Stop</TITLE>
        <ARTIST>Sam Brown</ARTIST>
        <COUNTRY>UK</COUNTRY>
        <COMPANY>A and M</COMPANY>
        <PRICE>8.90</PRICE>
        <YEAR>1988</YEAR>
    </CD>
    <CD>
        <TITLE>Bridge of Spies</TITLE>
        <ARTIST>T'Pau</ARTIST>
        <COUNTRY>UK</COUNTRY>
        <COMPANY>Siren</COMPANY>
        <PRICE>7.90</PRICE>
        <YEAR>1987</YEAR>
    </CD>
    <CD>
        <TITLE>Private Dancer</TITLE>
        <ARTIST>Tina Turner</ARTIST>
        <COUNTRY>UK</COUNTRY>
        <COMPANY>Capitol</COMPANY>
        <PRICE>8.90</PRICE>
        <YEAR>1983</YEAR>
    </CD>
    <CD>
        <TITLE>Midt om natten</TITLE>
        <ARTIST>Kim Larsen</ARTIST>
        <COUNTRY>EU</COUNTRY>
        <COMPANY>Medley</COMPANY>
        <PRICE>7.80</PRICE>
        <YEAR>1983</YEAR>
    </CD>
    <CD>
        <TITLE>Pavarotti Gala Concert</TITLE>
        <ARTIST>Luciano Pavarotti</ARTIST>
        <COUNTRY>UK</COUNTRY>
        <COMPANY>DECCA</COMPANY>
        <PRICE>9.90</PRICE>
        <YEAR>1991</YEAR>
    </CD>
    <CD>
        <TITLE>The dock of the bay</TITLE>
        <ARTIST>Otis Redding</ARTIST>
        <COUNTRY>USA</COUNTRY>
        <COMPANY>Atlantic</COMPANY>
        <PRICE>7.90</PRICE>
        <YEAR>1987</YEAR>
    </CD>
    <CD>
        <TITLE>Picture book</TITLE>
        <ARTIST>Simply Red</ARTIST>
        <COUNTRY>EU</COUNTRY>
```

```
		<COMPANY>Elektra</COMPANY>
		<PRICE>7.20</PRICE>
		<YEAR>1985</YEAR>
	</CD>
	<CD>
		<TITLE>Red</TITLE>
		<ARTIST>The Communards</ARTIST>
		<COUNTRY>UK</COUNTRY>
		<COMPANY>London</COMPANY>
		<PRICE>7.80</PRICE>
		<YEAR>1987</YEAR>
	</CD>
	<CD>
		<TITLE>Unchain my heart</TITLE>
		<ARTIST>Joe Cocker</ARTIST>
		<COUNTRY>USA</COUNTRY>
		<COMPANY>EMI</COMPANY>
		<PRICE>8.20</PRICE>
		<YEAR>1987</YEAR>
	</CD>
</CATALOG>
```

Introduction to XML

XML was designed to transport and store data.

HTML was designed to display data.

What You Should Already Know

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

- HTML
- JavaScript

If you want to study these subjects first, find the tutorials on our Home page.

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 is a W3C Recommendation

XML became a W3C Recommendation on February 10, 1998.

To read more about the XML activities at W3C, please read our W3C Tutorial.

XML is Everywhere

XML is now as important for the Web as HTML was to the foundation of the Web.

XML is the most common tool for data transmissions between all sorts of applications.

XML Tree

« Previous
Next Chapter »

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="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 is the XML declaration. It defines the XML version (1.0) and the encoding used (ISO-8859-1 = Latin-1/West European character set).

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.

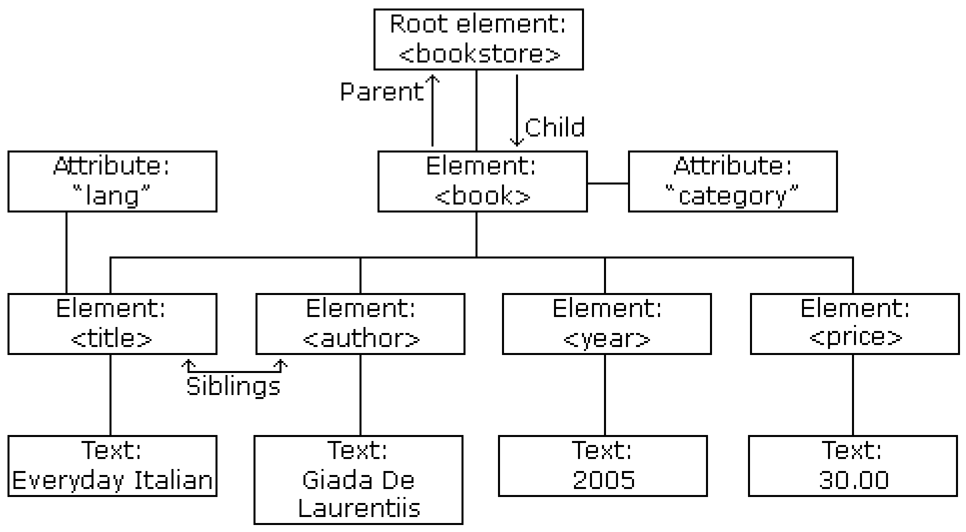
All elements can have sub elements (child elements):

```
<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 Rules

« Previous
Next Chapter »

The syntax rules of XML are very simple and logical. The rules are easy to learn, and easy to use.

All XML Elements Must Have a Closing Tag

In HTML, some elements do not have to have a closing tag:

```
<p>This is a paragraph.
<br>
```

In XML, it is illegal to omit the closing tag. All elements **must** have a closing tag:

```
<p>This is a paragraph.</p>
<br />
```

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, and it has no closing tag.

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:

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

Note: "Opening and closing tags" are often referred to as "Start and end tags". Use whatever you prefer. It is exactly the same thing.

XML Elements Must be Properly Nested

In HTML, you might see improperly nested elements:

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

In XML, all elements **must** be properly nested within each other:

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

In the example above, "Properly nested" simply means that since the <i> element is opened inside the <b> element, it must be closed inside the <b> element.

XML Documents Must Have a Root Element

XML documents must contain one element that is the **parent** of all other elements. This element is called the **root** element.

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

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.

Entity References

Some characters have a special meaning in XML.

If you place a character like "<" inside an XML element, it will generate an error because the parser interprets it as the start of a new element.

This will generate an XML error:

```
<message>if salary < 1000 then</message>
```

To avoid this error, replace the "<" character with an **entity reference**:

```
<message>if salary &lt; 1000 then</message>
```

There are 5 predefined entity references in XML:

&lt;	<	less than
&gt;	>	greater than
&amp;	&	ampersand
&apos;	'	apostrophe
&quot;	"	quotation mark

Note: Only the characters "<" and "&" are strictly illegal in XML. The greater than character is legal, but it is a good habit to replace it.

Comments in XML

The syntax for writing comments in XML is similar to that of HTML.

<!-- This is a comment -->

White-space is Preserved in XML

HTML truncates multiple white-space characters to one single white-space:

HTML:	Hello Tove
Output:	Hello Tove

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

XML Stores New Line as LF

In Windows applications, a new line is normally stored as a pair of characters: carriage return (CR) and line feed (LF). In Unix applications, a new line is normally stored as an LF character. Macintosh applications also use an LF to store a new line.

XML stores a new line as LF.

XML Elements

An XML document contains XML Elements.

What is an XML Element?

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.

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. Names with an underscore separator are nice: <first_name>, <last_name>.

Names should be short and simple, like this: <book_title> not like this: <the_title_of_the_book>.

Avoid "-" characters. If you name something "first-name," some software may think you want to subtract name from first.

Avoid "." characters. If you name something "first.name," some software may think that "name" is a property of the object "first."

Avoid ":" characters. Colons are reserved to be used for something called namespaces (more later).

XML documents often have a corresponding database. 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, but watch out for problems if your software vendor 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 Attributes

« Previous
Next Chapter »

XML elements can have attributes, just like HTML.

Attributes provide additional information about an element.

XML Attributes

In HTML, 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 can be important to the software that wants to manipulate the element:

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

XML Attributes Must be Quoted

Attribute values must always be quoted. Either single or double quotes can be used. For a person's sex, the person element can be written like this:

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

or like this:

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

If the attribute value itself contains double quotes you can use single quotes, like in this example:

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

or you can use character entities:

```
<gangster name="George &quot;Shotgun&quot; Ziegler">
```

XML Elements vs. 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 an element. Both examples provide the same information.

There are no rules about when to use attributes or when to use elements. Attributes are handy in HTML. In XML my advice is to avoid them. Use elements instead.

My Favorite Way

The following three XML documents contain exactly the same information:

A date attribute is used in the first example:

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

Avoid XML Attributes?

Some of the problems with using attributes are:

- attributes cannot contain multiple values (elements can)
- attributes cannot contain tree structures (elements can)
- attributes are not easily expandable (for future changes)

Attributes are difficult to read and maintain. Use elements for data. Use attributes for information that is not relevant to the data.

Don't end up like this:

```
<note day="10" month="01" year="2008"
to="Tove" from="Jani" heading="Reminder"
body="Don't forget me this weekend!">
</note>
```

XML Attributes for Metadata

Sometimes ID references are assigned to elements. These IDs can be used to identify XML elements in much the same way as the id attribute in HTML. This example demonstrates this:

```
<messages>
  <note id="501">
    <to>Tove</to>
    <from>Jani</from>
    <heading>Reminder</heading>
    <body>Don't forget me this weekend!</body>
  </note>
  <note id="502">
    <to>Jani</to>
    <from>Tove</from>
    <heading>Re: Reminder</heading>
    <body>I will not</body>
  </note>
</messages>
```

The id attributes above are for identifying the different notes. It is not a part of the note itself.

What I'm trying to say here is that metadata (data about data) should be stored as attributes, and the data itself should be stored as elements.