

Lecture 7

XML Tutorial [1]

Samkeun Kim <skim@hknu.ac.kr>

<http://cyber.hknu.ac.kr/>

XML Basic

<https://www.w3schools.com/xml/>

XML History

1970년대 중반, 찰스 F. 골드팔:

SGML (Standard Generalized Markup Language)

데이터 표현의 국제적 기준 제시

1980년대 중반, 팀 버너스-리:

World Wide Web 고안, W3C 결성

W3C 첫 번째 활동

HTML (Hypertext Markup Language) Spec.

형식과 문서 텍스트의 레이아웃 설명 체계 제공

1990년대 후반, e-비즈니스 발전

표준 비즈니스 중심의 데이터 표현 형식 요구.

For example, **XML** (Extensible Markup Language)

메타 정보(자기 묘사가 가능한 태그) 제공

XML

XML stands for e**X**tensible **M**arkup **L**anguage.

XML is designed to describe data.

XML is a software- and hardware-independent tool for carrying information.

XML is easy to learn.

```
<?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>
```

[Display this XML in Your Browser >>](#)

[Display Another >>](#)

[With Style >>](#)

Introduction to XML

XML was designed to describe 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

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 describe 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.

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

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.

How Can XML be **Used?**

XML is used in **many aspects of web development**, often to simplify data storage and sharing.

XML Separates Data from HTML

If you need to **display dynamic data** in your HTML document, it will take a lot of work to **edit the HTML each time the data changes**.

With XML, data can be **stored in separate XML files**:

This way you can concentrate on using HTML/CSS for display and layout, and be sure that **changes in the underlying data will not require any changes to the HTML**

With a few lines of JavaScript code, you can read an external XML file and update the data content of your web page.

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 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, 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 Makes Your Data More Available

Different applications can access your data, not only in HTML pages, but also from XML data sources.

With XML, your data can be **available to all kinds of "reading machines"** (Handheld computers, voice machines, news feeds, etc).

Internet Languages Written in XML

Several Internet languages are written in XML.

Here are some examples:

- [XHTML](#)
- [XML Schema](#)
- [RDF](#)
- [WSDL](#)
- [RSS](#)

XML Tree

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

An Example XML Document (1/2)

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) and **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

"Self-describing"!!

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):

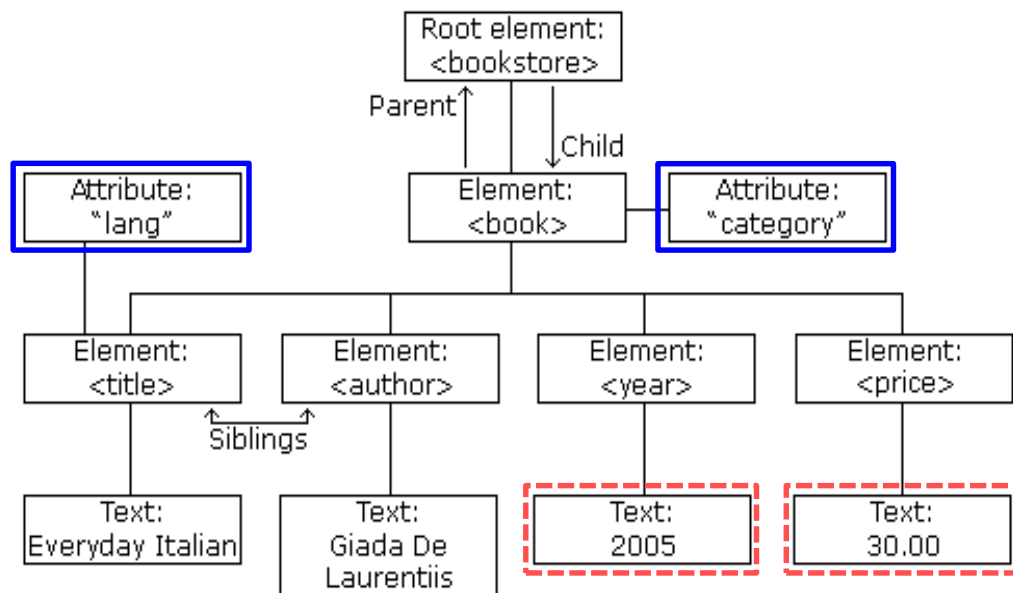
```
<root>  
  <child>  
    <subchild>.....</subchild>  
  </child>  
</root>
```

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**



The image above represents **one book** in the XML below:

- 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>`**

```
<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>
```

XML Syntax

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>
```

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>
```

"Properly nested" simply means that since the `<i>` element is opened inside the `` element, it **must be closed inside** the `` 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

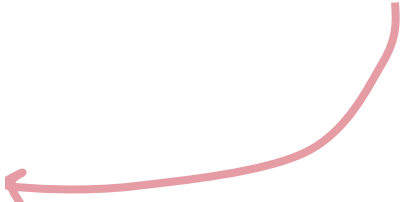
XML elements can have attributes in name/value pairs just like in HTML

In XML, **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>
```

Entity References (1/2)

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**:

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

This will generate an XML **error**.

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

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


Entity References (2/2)

There are **5 predefined entity references** in XML:

<	<	less than
>	>	greater than
&	&	ampersand
'	'	apostrophe
"	"	quotation mark

Comments in XML

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? (1/2)

An XML element is everything from 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...**

What is an XML Element? (2/2)

```
<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>
```

<bookstore> and <book> have element **contents**.

<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 />
```

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>`.

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 vendor doesn't support them

Naming Styles

There are **no naming styles defined for XML elements**.

But here are some commonly used:

Style	Example	Description
Lower case	<firstname>	All letters lower case
Upper case	<FIRSTNAME>	All letters upper case
Underscore	<first_name>	Underscore separates words
Pascal case	<FirstName>	Uppercase first letter in each word
Camel case	<firstName>	Uppercase first letter in each word except the first

If you choose a naming style, it is good to be **consistent!**

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.

XML Elements are Extensible (1/2)

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!

XML Elements are Extensible (2/2)

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

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:

```
  
<a href="demo.asp">
```

Attributes often provide information that is not a part of the data

In the example below, **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>
```

*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

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>
```

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>
```

Three XML documents contain
exactly the same information

expanded date element is used in the third:

```
<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

- ✓ **IDs can be used to identify XML elements** in much the same way as the id attribute in HTML.

```
<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.

Metadata (data about data) should be stored as attributes,
and the data itself should be stored as elements.

XML Namespaces

XML Namespaces provide a **method to avoid element name conflicts**.

Name Conflicts

In XML, element names are **defined by the developer**:

- ✓ This often **results in a conflict when trying to mix XML documents from different XML applications**

This XML carries **HTML table** information:

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

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

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

If these XML fragments were **added together**, there would be a name conflict.

Both contain a <table> element, but the elements have different content and meaning.

A user or an **XML application will not know how to handle these differences.**

Solving the Name Conflict Using a Prefix

Name conflicts in XML **can easily be avoided using a name prefix.**

This XML carries information about an HTML table, and a piece of furniture:

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

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

Will be **no conflict** because the two <table> elements have different names.

XML Namespaces - xmlns Attribute

When using prefixes in XML, a so-called **namespace for the prefix must be defined**.

Namespace is defined by xmlns attribute in the start tag of an element.

Syntax: **xmlns:prefix="URI"**

```
<root>

<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>

<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>

</root>
```

xmlns attribute in the <table> tag give the h: and f: prefixes a **qualified namespace**.

All child elements with the same prefix are associated with the same namespace

```
<root xmlns:h="http://www.w3.org/TR/html4/"
xmlns:f="http://www.w3schools.com/furniture">

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

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

</root>
```

NOTE Namespace **URI** is not used by the parser to look up information. The purpose is to give the namespace a unique name. Try to go to <http://www.w3.org/TR/html4/>

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.

Default Namespaces

Defining a **default namespace** for an element saves us from using prefixes in all the child elements

```
xmlns="namespaceURI"
```

This XML carries HTML table information:

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

This XML 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>
```


Namespaces in Real Use

XSLT: XML language that can be used to **transform XML documents into other formats**

Tags that are not HTML tags have the prefix **xsl**, identified by the namespace **xmlns:xsl="http://www.w3.org/1999/XSL/Transform"**:

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

<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

  <xsl:template match="/">
    <html>
    <body>
      <h2>My CD Collection</h2>
      <table border="1">
        <tr>
          <th style="text-align:left">Title</th>
          <th style="text-align:left">Artist</th>
        </tr>
        <xsl:for-each select="catalog/cd">
          <tr>
            <td><xsl:value-of select="title"/></td>
            <td><xsl:value-of select="artist"/></td>
          </tr>
        </xsl:for-each>
      </table>
    </body>
    </html>
  </xsl:template>

</xsl:stylesheet>
```

XML Encoding

XML documents can contain international characters, like Norwegian æøå, or French êëé.

To avoid errors, you should 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, code set, 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 1 byte (8-bits) to represent characters in the **ASCII set**, and two or three bytes for the rest.

UTF-16 uses 2 bytes (16 bits) for most characters, and four bytes for the rest.

UTF-8 - Web Standard

UTF-8 is the standard character encoding on the web.

UTF-8 is the default character encoding for HTML5, CSS, JavaScript, PHP, SQL, and XML.

XML Encoding

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"?>
```

UTF-8 is the default for documents without encoding information.

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](#) and [Note saved with wrong encoding](#).

For high quality XML documents, **UTF-8 encoding is the best to use**.

UTF-8 covers international characters, and it is also the default, if no encoding is declared.

Viewing XML Files

Raw XML files can be viewed in all major browsers.

Don't expect XML files to be displayed as HTML pages.

Viewing XML Files

```
<?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>
```

Look at this XML file: [note.xml](#)

- ✓ XML document will be displayed 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
- ✓ To view the raw XML source (without the + and - signs), select "View Page Source" or "View Source" from the browser menu

Viewing an Invalid XML File

If an erroneous XML file is opened, some browsers might report the error, some may display it incorrectly.

Try to open this XML file in Chrome, IE, Firefox, Opera, and Safari : [note_error.xml](#).

Other XML Examples

Viewing some XML documents will help you get the XML feeling.

[An XML CD catalog](#)

This is a **CD collection**, stored as XML data.

[An XML plant catalog](#)

This is a **plant catalog** from a plant shop, stored as XML data.

[A Simple Food Menu](#)

This is a **breakfast food menu** from a restaurant, stored as XML data.

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.

Displaying XML with CSS

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

Displaying your XML Files with CSS?

It is possible to use CSS to format an XML document.

Below is an example of how to use a CSS style sheet to format an XML document:

- ✓ Take a look at this XML file: [CD catalog](#)
- ✓ Then look at this style sheet: [CSS file](#)
- ✓ Finally, view: [CD catalog formatted with the CSS file](#)

Second line links XML file to CSS file:

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <?xml-stylesheet type="text/css" href="cd_catalog.css"?>
3 <CATALOG>
4   <CD>
5     <TITLE>Empire Burlesque</TITLE>
6     <ARTIST>Bob Dylan</ARTIST>
7     <COUNTRY>USA</COUNTRY>
8     <COMPANY>Columbia</COMPANY>
9     <PRICE>10.90</PRICE>
10    <YEAR>1985</YEAR>
11  </CD>
12  <CD>
13    <TITLE>Hide your heart</TITLE>
14    <ARTIST>Bonnie Tyler</ARTIST>
15    <COUNTRY>UK</COUNTRY>
16    <COMPANY>CBS Records</COMPANY>
17    <PRICE>9.90</PRICE>
18    <YEAR>1988</YEAR>
19  </CD>
20  <CD>
21    <TITLE>Greatest Hits</TITLE>
22    <ARTIST>Dolly Parton</ARTIST>
23    <COUNTRY>USA</COUNTRY>
24    <COMPANY>RCA</COMPANY>
25    <PRICE>9.90</PRICE>
26    <YEAR>1982</YEAR>
27  </CD>
28 ...
```

cd_catalog.css

```
1 CATALOG {
2   background-color: #ffffff;
3   width: 100%;
4 }
5 CD {
6   display: block;
7   margin-bottom: 30pt;
8   margin-left: 0;
9 }
10 TITLE {
11   display: block;
12   color: #ff0000;
13   font-size: 20pt;
14 }
15 ARTIST {
16   display: block;
17   color: #0000ff;
18   font-size: 20pt;
19 }
20 COUNTRY, PRICE, YEAR, COMPANY {
21   display: block;
22   color: #000000;
23   margin-left: 20pt;
24 }
```

Formatting XML with CSS is not the most common method.

W3C recommends using **XSLT** instead.

실습과제 07-1

한 서점이 확보하고 있는 모든 도서 정보를 브라우저에 표시해주는 **bookstore.xml** 문서를 작성하시오.

또한 **CSS**를 이용하여 각 도서에 대한 정보를 브라우저에 보기 좋게 디스플레이하시오. 단, 해당 서점은 10권 이상의 도서를 확보하고 있다고 가정하시오.

Q & A

