**XML**

XML stands for eXtensible Markup Language.

**XML was designed to store and transport data.**

XML was designed to be both human- and machine-readable.

XML Example 1

**<?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 Difference Between XML and HTML

XML and HTML were designed with different goals:

* XML was designed to carry data - with focus on what data is
* HTML was designed to display data - with focus on how data looks
* XML tags are not predefined like HTML tags are

## XML Usage

A short list of XML usage says it all −

* XML can work behind the scene to simplify the creation of HTML documents for large web sites.
* XML can be used to exchange the information between organizations and systems.
* XML can be used for offloading and reloading of databases.
* XML can be used to store and arrange the data, which can customize your data handling needs.
* XML can easily be merged with style sheets to create almost any desired output.
* Virtually, any type of data can be expressed as an XML document.

## Key Features of XML

Self describing data When you look at an XML document, it is very easy to describe data in table structure format.

Creating custom tag XML language easy to creating your defined tag to describing data.

Exchanging data XML data you can sharing easily between different application as well as database.

## What is Markup?

XML is a markup language that defines set of rules for encoding documents in a format that is both human-readable and machine-readable. So *what exactly is a markup language?* Markup is information added to a document that enhances its meaning in certain ways, in that it identifies the parts and how they relate to each other. More specifically, a markup language is a set of symbols that can be placed in the text of a document to demarcate and label the parts of that document.

Following example shows how XML markup looks, when embedded in a piece of text −

<message>

<text>Hello, world!</text>

</message>

This snippet includes the markup symbols, or the tags such as <message>...</message> and <text>... </text>. The tags <message> and </message> mark the start and the end of the XML code fragment. The tags <text> and </text> surround the text Hello, world!.

## Is XML a Programming Language?

A programming language consists of grammar rules and its own vocabulary which is used to create computer programs. These programs instruct the computer to perform specific tasks. XML does not qualify to be a programming language as it does not perform any computation or algorithms. It is usually stored in a simple text file and is processed by special software that is capable of interpreting XML.

|  |  |  |
| --- | --- | --- |
| Key Point | XML | HTML |
| stands for | eXtensible Markup Language | Hyper Text Markup Language |
| Derived from | XML derived from SGML(Standard Generalized Markup Language). | Where as the HTML derived same from SGML. |
| Purpose | XML was designed for **holds data.** Use for **transport data** between **application and database**. | HTML was designed for **specify how to data**should be **display on web page**. |
| Rules | XML was follow **strict rules**. Any time terminate the process if rules break. | HTML was **not following any strict rules**. All browser try to display data to the best as per its ability. |
| Case Sensitive | XML is **case sensitive**. | HTML is **not a case sensitive**. |
| Uses | XML uses for **temporary/permanent storing data**. But now a day use for write markup language that are render/connect to a application and database. | HTML uses for **web presentation layer** along with CSS you can make very interactive design. |
| Tags | XML tags you can **define custom tag** by ourself. | HTML tags are **predefined**. |
| How to write | XML tags must have **closing tag**. Example. <note>Travel experience</note> | HTML tags are two type **closing tag** or **self-closing tag**. Example. **self-closing tag** <br />, **Closing tag** <p>Travel experience</p> |
| Whitespace | XML was preserve only **one whitespace**. | Where as the HTML was preserve only **one whitespace**. |
| Behavior | XML was **dynamic** for holding data. | While HTML was **static** for displaying data. |

# XML Syntax Rules

## XML Documents Must Have a Root Element

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

**<root>  
  <child>  
    <subchild>.....</subchild>  
  </child>  
</root>**

In this example **<note>** is the root element:

<?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>

## All XML Elements Must Have a Closing Tag

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

<p>This is a paragraph.</p>  
<br />

## 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 correct</message>

"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.

## What is an XML Schema?

An XML Schema describes the structure of an XML document.

The XML Schema language is also referred to as XML Schema Definition (XSD).

An XML Schema is a language for expressing constraints about XML documents. There are several different schema languages in widespread use, but the main ones are Document Type Definitions (DTDs), Relax-NG, Schematron and W3C XSD (XML Schema Definitions).

The purpose of an XML Schema is to define the legal building blocks of an XML document:

* the elements and attributes that can appear in a document
* the number of (and order of) child elements
* data types for elements and attributes
* default and fixed values for elements and attributes

# XML DTD

An XML document with correct syntax is called "Well Formed".

An XML document validated against a DTD is both "Well Formed" and "Valid".

## Valid XML Documents

A "Valid" XML document is a "Well Formed" XML document, which also conforms to the rules of a DTD:

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

The DOCTYPE declaration, in the example above, is a reference to an external DTD file. The content of the file is shown in the paragraph below.

## XML DTD

The purpose of a DTD is to define the structure of an XML document. It defines the structure with a list of legal elements:

**<!DOCTYPE note  
[  
<!ELEMENT note (to,from,heading,body)>  
<!ELEMENT to (#PCDATA)>  
<!ELEMENT from (#PCDATA)>  
<!ELEMENT heading (#PCDATA)>  
<!ELEMENT body (#PCDATA)>  
]>**

The DTD above is interpreted like this:

* !DOCTYPE note defines that the root element of the document is note
* !ELEMENT note defines that the note element must contain the elements: "to, from, heading, body"
* !ELEMENT to defines the to element to be of type "#PCDATA"
* !ELEMENT from defines the from element to be of type "#PCDATA"
* !ELEMENT heading defines the heading element to be of type "#PCDATA"
* !ELEMENT body defines the body element to be of type "#PCDATA"

## What is a DTD?

A DTD is a Document Type Definition.

A DTD defines the structure and the legal elements and attributes of an XML document.

## Why Use a DTD?

With a DTD, independent groups of people can agree on a standard DTD for interchanging data.

An application can use a DTD to verify that XML data is valid.

## An Internal DTD Declaration

If the DTD is declared inside the XML file, it must be wrapped inside the <!DOCTYPE> definition:

### XML document with an internal DTD

**<?xml version="1.0"?>  
<!DOCTYPE note [  
<!ELEMENT note (to,from,heading,body)>  
<!ELEMENT to (#PCDATA)>  
<!ELEMENT from (#PCDATA)>  
<!ELEMENT heading (#PCDATA)>  
<!ELEMENT body (#PCDATA)>  
]>  
<note>  
<to>Tove</to>  
<from>Jani</from>  
<heading>Reminder</heading>  
<body>Don't forget me this weekend</body>  
</note>**

The DTD above is interpreted like this:

* **!DOCTYPE note** defines that the root element of this document is note
* **!ELEMENT note** defines that the note element must contain four elements: "to,from,heading,body"
* **!ELEMENT to** defines the to element to be of type "#PCDATA"
* **!ELEMENT from** defines the from element to be of type "#PCDATA"
* **!ELEMENT heading** defines the heading element to be of type "#PCDATA"
* **!ELEMENT body** defines the body element to be of type "#PCDATA"

## An External DTD Declaration

If the DTD is declared in an external file, the <!DOCTYPE> definition must contain a reference to the DTD file:

### XML document with a reference to an external DTD

<?xml version="1.0"?>  
**<!DOCTYPE note SYSTEM "note.dtd">**<note>  
  <to>Tove</to>  
  <from>Jani</from>  
  <heading>Reminder</heading>  
  <body>Don't forget me this weekend!</body>  
</note>

And here is the file "note.dtd", which contains the DTD:

<!ELEMENT note (to,from,heading,body)>  
<!ELEMENT to (#PCDATA)>  
<!ELEMENT from (#PCDATA)>  
<!ELEMENT heading (#PCDATA)>  
<!ELEMENT body (#PCDATA)>

**DOM:** DOM is an interface-oriented Application Programming Interface. It allows for

navigation of the entire document. A DOM document can be created by a parser, or can be

generated manually by users. Data types in DOM Nodes are abstract. DOM implementations

generally require the entire document to be loaded into memory and constructed as a tree of

objects before access is allowed. DOM is supported in Java. Its specification is regulated by the

World Wide Web Consortium.

**Difference between DOM and SAX.**

• SAX parser works incrementally and generates events that are passed to the application.

DOM parser reads the whole XML document and returns a DOM tree representation of

xml document

• In DOM the xml file is arranged as a tree and backward and forward search is possible In

SAX traversing in any direction is not possible as Top to bottom approach is used.

• SAX is essentially an API for reading XML, and not writing it. DOM allows you to read

and write.

**Features of DOM**

DOM is Document Object Model. It is used to read data from a XML document. It is more

commonly used in applications where data in the document needs to be repeated accessed. DOM

supports navigation in any direction. XML DOM is typically used for XML documents. The

DOM defines the objects and properties of all document elements, and the methods (interface) to

access them.

**Features of SAX:**

SAX is Simple API for XML. It is used to read data from a XML document. A parser that uses

SAX parses the XML serially. The API is event driven and these events are fired when the XML features are encountered. XML parsing is unidirectional. Memory used by a SAX parses is

relatively low. Due to the event nature of SAX, the parsing is faster of an XML document.

**XML Schema**

* 1. The purpose of an **XML Schema** is to define the legal building blocks of an XML document: a. the elements and attributes that can appear in a document

the number of (and order of) child elements

data types for elements and attributes

default and fixed values for elements and attributes