XML:

**What is XML?**

XML is a markup language which is used for storing and transporting data. XML doesn’t depend on the platform and the software(programming language). You can write a program in any language on any platform (Operating System) to send, receive or store data using XML.

## XML Example

## XML Example

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

<message>

<to>MyReader</to>

<from>Chaitanya</from>

<msg>Welcome to beginnersbook.com</msg>

</message>

## XML Properties

1. XML is a markup language that focuses on data rather than how the data looks.  
2. XML is designed to send, store, receive and display data. In simple words you can say that XML is used for storing and transporting data.  
3. XML became a W3C (W3C stands for World Wide Web Consortium, the main international standards organization for the World Wide Web) recommendation on February 10, 1998.  
4. XML is different from HTML. XML focuses on data while HTML focuses on how the data looks.  
5. XML does not depend on software and hardware, it is platform and programming language independent.  
6. Unlike HTML where most of the tags are predefined, XML doesn’t have predefined tags, rather you have to create your own tags.

## Why we need XML?

Since there are systems with different-different operating systems having data in different formats. In order to transfer the data between these systems is a difficult task as the data needs to converted in compatible formats before it can be used on other system. With XML, it is so easy to transfer data between such systems as XML doesn’t depend on platform and the language

## Features of XML

### 1. XML focuses on data rather than how it looks

One of the reason, XML is popular because it focuses on data rather than data presentation. The other markup language such as HTML is used for data presentation. This separates the data and its presentation part and gives us the freedom to present the data, the way we want, once we receive it using XML.

Two or more systems can receive the same data from a same XML and present it in a different way using other markup language such as HTML.

### 2. Easy and efficient data sharing

Since XML is **software and hardware independent**, it is easier to share data between different systems with different hardware and software configuration. Any system with any programming language can read and process a XML document.

### 3. Compatibility with other markup language HTML

It is so much easier to read the data from XML and display it on an GUI(graphical user interface) using HTML markup language.

When the data changes over time, we need not to make any changes in the HTML.

### 4. Supports platform transition

The main reason why changing to new systems and platform is challenging, because it involves the headache of data conversion between incompatible formats which often results in data loss. XML simplifies this process as the data is transported on new upgraded systems without any data loss.

### 5. Allows XML validation

A XML document can be validated using DTD or XML schema. This ensures that the XML document is syntactically correct and avoids any issues that may arise due to the incorrect XML.

### 6. Adapts technology advancements

The reason why XML is popular and being used from a very long time is because, it can adapt to the new technologies because of its platform-independent nature.

### 7. XML supports Unicode

XML supports Unicode that allows it to communicate almost any information in any written human language.

# HTML vs XML

BY CHAITANYA SINGH | FILED UNDER: [**XML TUTORIAL**](https://beginnersbook.com/category/xml-tutorial/)

In the previous tutorial of [**XML introduction**](https://beginnersbook.com/2018/10/xml-introduction/), we discussed the **differences between HTML and XML** in brief. In this guide, we will discuss them in detail.

|  |  |  |
| --- | --- | --- |
| **SNo.** | **HTML(Hyper Text Markup Language)** | **XML(eXtensible Markup Language)** |
| 1 | HTML focuses on how the data looks | XML focuses on the data rather than how it looks |
| 2 | HTML is **not a case sensitive language** | XML is **case sensitive language** |
| 3 | HTML is mainly concerned with the presentation of data | XML is mainly used for storing and transporting the data |
| 4 | HTML is static | XML is dynamic |
| 5 | In HTML the closing tag in optional | In XML the closing tag is mandatory |
| 6 | HTML uses predefined tags such as <b>, <br>, <img> etc. | XML uses the user-defined tags that we create while writing the XML document. |
| 7 | HTML does not preserve white space. | XML preserves white space. |

# JSON vs XML

BY CHAITANYA SINGH | FILED UNDER: [**XML TUTORIAL**](https://beginnersbook.com/category/xml-tutorial/)

In this guide, we will learn the **differences and similarities between**[**JSON**](https://beginnersbook.com/2015/04/json-tutorial/)**and**[**XML**](https://beginnersbook.com/2018/10/xml-introduction/).

Before we see the differences between XML and JSON, lets see the example of a JSON file and the same data in XML file.

## JSON Example

A simple example of a student array data in JSON format.

{"student":[

{"name":"Steve", "nickname":"Stevie"},

{"name":"Chaitanya", "nickname":"Chetan"},

{"name":"Ajeet", "nickname":"Jaat"},

{"name":"Lina", "nickname":"Lee"}

]}

## XML Example

The same above example in XML document.

<students>

<student>

<name>Steve</name>

<nickname>Stevie</nickname>

</student>

<student>

<name>Chaitanya</name>

<nickname>Chetan</nickname>

</student>

<student>

<name>Ajeet</name>

<nickname>Jaat</nickname>

</student>

<student>

<name>Lina</name>

<nickname>Lee</nickname>

</student>

</students>

## JSON vs XML

|  |  |  |
| --- | --- | --- |
| **SNO.** | **JSON (JavaScript Object Notation)** | **XML (eXtensible Markup Language)** |
| 1 | JSON is simple and easier to read and write | XML is verbose and less readable |
| 2 | JSON doesn’t use end tag | In XML, the end tag is mandatory |
| 3 | JSON supports array thus it is easy to transfer a big chunk of homogeneous data items using JSON | XML doesn’t support array |
| 4 | JSON is easier to parse and can be parsed to ready-to-use JavaScript object | XML is difficult to parse than JSON |
| 5 | JSON is short | XML document is lengthy, verbose and redundant |
| 6 | JSON is less secure than XML | XML is more secured than JSON |
| 7 | JSON file is more readable than XML because it is short and to the point. | XML file is big and filled with user-defined tags, thus less-readable |
| 8 | JSON is data-oriented | XML is document-oriented |

## Simple XML document example

This is a simple XML document. You can understand the data, by just looking at the document. We have used self-describing tags.

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

<book>

<name>A Song of Ice and Fire</name>

<author>George R. R. Martin</author>

<language>English</language>

<genre>Epic fantasy</genre>

</book>

**The first line** <?xml version="1.0" encoding="UTF-8"?> is called **XML Prolog**. It is optional, however when we include it in the XML document, it should always be the first line of the document. XML Prolog defines the XML version and the encoding used in the XML document.

**The tag** <book> is the **root of this XML document**. A XML document **should always have a root element** and at the end of the document this root element needs a closing tag, just like </book> that we have used in the above example.

The tags <name>, <author>, <language> and <genre> are the child elements of the root element <book>. We will discuss more about these in the XML syntax tutorial.

Based on the above discussion we can say that a XML document structure looks like this:

<root>

<child>

<subchild>.....</subchild>

</child>

</root>

## XML Example – Student data

Lets take a look at the another example of XML. In this XML document we have the details of the few students. Here <students> is the root element, <student> is the child element and name, age, subject and gender are sub-child elements.

<students>

<student>

<name>Rick Grimes</name>

<age>35</age>

<subject>Maths</subject>

<gender>Male</gender>

</student>

<student>

<name>Daryl Dixon </name>

<age>33</age>

<subject>Science</subject>

<gender>Male</gender>

</student>

<student>

<name>Maggie</name>

<age>36</age>

<subject>Arts</subject>

<gender>Female</gender>

</student>

</students>

## 1. Root Element is mandatory in XML

XML document **must have** a root element. A root element can have child elements and sub-child elements.

For example: In the following XML document, <message> is the root element and <to>, <from>, <subject> and <text> are child elements.

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

<message>

<to>Steve</to>

<from>Paul</from>

<subject>Message from teacher to Student</subject>

<text>You have an exam tomorrow at 9:00 AM</text>

</message>

The **following XML document is wrong** because it doesn’t have a root element.

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

<to>Steve</to>

<from>Paul</from>

<subject>Message from teacher to Student</subject>

<text>You have an exam tomorrow at 9:00 AM</text>

## 2. XML is case sensitive

XML is a case sensitive language.

For example:  
**This is valid**

<from>Paul</from>

**This is invalid**  
The first letter of closing tag is in capital while the first letter of opening tag is in small, this is an [**example**](https://beginnersbook.com/2018/10/xml-example/) of invalid XML.

<from>Paul</From>

## 3. XML Prolog

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

This line is called the XML Prolog. It is an optional line, however it should be the first line when you mention it. It specifies the XML version and the encoding used in the XML document.

## 4. Elements should not overlap

All the elements in XML should be properly nested and they should not overlap.

<class><teacher>Rick</class></teacher> -->Wrong (Not nested properly)

<class><teacher>Rick</teacher></class> -->Correct (Correctly nested)

## 5. Attributes in XML

We will discuss XML attributes in detail later. For now, lets see the syntax of attributes. An opening tag in XML can have attributes, these attributes are name & value pairs.

Attribute names are **case sensitive** and should **not be in quotation** marks.  
Attribute values should **be in single or double quotation**.

<text category = "message">You have an exam tomorrow at 9:00 AM</text>

Here category is the attribute name and message is the attribute value.

Lets take few more examples to see valid and invalid cases of attributes.  
A tag can have more than one name & value pairs, however two attribute names cannot be same (see point 5 in the following example)

1. <text category = message>hello</text> -->wrong

2. <text "category" = message>hello</text> -->wrong

3. <text category = "message">hello</text> -->correct

4. <text category = "message" purpose = "greet">hello</text> -->correct

5. <text category = "message" category ="greet">hello</text> -->wrong

## 6. XML elements must have a closing tag

All XML documents must have a closing tag.

<text category = message>hello</text> -->correct

<text category = message>hello -->wrong

## 7. Comments in XML

This is how a comment should look like in XML document.

<!-- This is just a comment -->

## XML Tree

XML document has a tree structure, where the root element is at the top and the child elements are connected to the root elements, the same way, how leaves are connected to tree through branches. We will first see an example of XML document and then we will draw a tree structure based on the example.

### Example of XML document

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

<company>

<employee>

<name>Negan</name>

<age>40</age>

<email>imnegan@twd.com</email>

<address>

<city>Noida</city>

<state>Uttar Pradesh</state>

<pin>201301</pin>

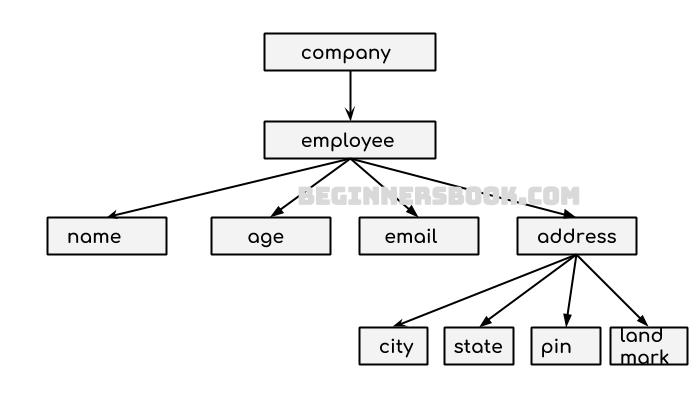
<landmark>Near hill top</landmark>

</address>

</employee>

</company>

### Tree Structure

The tree structure of the above XML document would look like this:  


**Root element**: <company>  
**Child elements**: <name>, <age>, <email> & <address>  
**Sub-child elements**: <city>, <state>, <pin> & <landmark>

## What are XML attributes?

XML attributes are a way to add additional data to the XML element. Attributes contain data in form of name & value pairs.

**For example:** Here we have added an attribute name brand and value Ford Figo to the element <car>

<car brand="Ford Figo">

<name>Ford Figo 1.2P Ambiente MT</name>

<horsepower>65 kW @ 6,300 rpm</horsepower>

</car>

## XML example with more than one attributes

In the above example we have a single attribute associated with the element <car>, however we can have more than one attribute associated to the same element as well. Lets take an example –

<car brand="Ford Figo" category="Subcompact car">

<name>Ford Figo 1.2P Ambiente MT</name>

<horsepower>65 kW @ 6,300 rpm</horsepower>

</car>

In the above example brand and category are the attributes associated with element car.

## Rules for XML attributes

Some of the rules related to the **XML attributes** have already been discussed in the [**XML syntax**](https://beginnersbook.com/2018/10/xml-syntax/) guide, however lets recall those rules here along with some other rules.

1. Attributes are name & value pairs. The attribute name should not be in quotes, however the attribute value must always be in quotes (single or double)

<car brand="Ford Figo"> --> Correct

<car "brand"=Ford Figo> --> Incorrect

2. XML element can have more than one attributes. This we have already seen in the above example, where brand and category attributes are linked to the element <car>.

3. Attributes cannot contain duplicate multiple values.

<car brand="Ford Figo" brand="Subcompact car"> -->wrong

## XML attributes vs XML Elements

Lets have a look at the following two examples –

**Example 1: Here brand is an attribute**

<car brand="Ford Figo">

<name>Ford Figo 1.2P Ambiente MT</name>

<horsepower>65 kW @ 6,300 rpm</horsepower>

</car>

**Example 2: Here brand is an element**

<car>

<brand>Ford Figo</name>

<name>Ford Figo 1.2P Ambiente MT</name>

<horsepower>65 kW @ 6,300 rpm</horsepower>

</car>

In both the examples, the XML document contain the same data. However you should always prefer XML element over XML attribute. Why? Lets discuss it in the following section.

## Why you should prefer XML element over attribute?

In the above two examples, we learned that the same information can be contained in a XML document by using XML element in place of attribute. The reason we do this is because attributes have certain limitations in XML –

1. attributes cannot have multiple values while elements can.  
2. Using attributes we cannot achieve the [**XML tree structure**](https://beginnersbook.com/2018/10/xml-tree-structure/) that we can achieve using element and sub-elements.  
3. Attributes values are difficult to test against a DTD  
4. Elements are easy to be handled by the programming language compared to the attributes

## Syntax of XML comment

<!-- This is just a comment. It is optional-->

Whatever written between <!-- and --> will be treated as a comment.

# XML Validator

BY CHAITANYA SINGH | FILED UNDER: [**XML TUTORIAL**](https://beginnersbook.com/category/xml-tutorial/)

The cool thing about XML is that it can be validated for syntax error using DTD or Schema. In this guide we will learn what is a valid xml, what are the rules and the ways to check for a valid XML document.

## Well Formed XML document

An XML document with correct syntax is known as valid XML document. Lets see few important rules to check for syntax errors.

**Rules:**  
1. All XML documents must have a root element.  
2. XML is a case sensitive language so you should be careful with the case while opening and closing tags.  
3. All XML tags must have a closing tag.  
4. XML attribute name should not be quoted while its value must be quoted.

For more rules check this guide: [**XML Syntax**](https://beginnersbook.com/2018/10/xml-syntax/)

## Ways to check a valid XML document

There are two document type definitions that can used with XML document to check whether the XML document is valid.  
1. XML DTD (Document Type Definition)  
2. XML Schema – An XML-based alternative to the Document Type Definition

### 1. XML DTD

DTD defines the structure of XML document that can be validated against the XML document to check for the syntax errors. XML DTD defines the structure by mentioning the XML elements in such a way so that the complete structure of XML document can be understood. The DTD file has .dtd extension.

### 2. XML Schema

The Schema does the same thing that a DTD can do. It also defines the structure of the XML document but unlike DTD it is an XML file, in addition to that Schema supports data types and namespaces.

In the [**previous tutorial**](https://beginnersbook.com/2018/10/xml-validator/), we learned that there are two ways to validate an XML, one of them is XML DTD that can be used to check whether an XML document is “well formed” and “valid”. In this guide, we will learn what is an **XML DTD** with the help of few examples.

## XML DTD

DTD stands for Document Type Definition. An XML DTD defines the structure of an XML document. An XML document is considered “well formed” and “valid” if it is successfully validated against DTD.

## An example of DTD

DTD is declared inside<!DOCTYPE> definition when the DTD declaration is internal. In this example we can see that there is an XML document that has a <!DOCTYPE> definition. The **bold part in the following example is the DTD declaration**.

<?xml version="1.0"?>

**<!-- XML DTD declaration starts here -->**

**<!DOCTYPE beginnersbook [**

**<!ELEMENT beginnersbook (to,from,subject,message)>**

**<!ELEMENT to (#PCDATA)>**

**<!ELEMENT from (#PCDATA)>**

**<!ELEMENT subject (#PCDATA)>**

**<!ELEMENT message (#PCDATA)>**

**]>**

**<!-- XML DTD declaration ends here-->**

<beginnersbook>

<to>My Readers</to>

<from>Chaitanya</from>

<subject>A Message to my readers</subject>

<message>Welcome to beginnersbook.com</message>

</beginnersbook>

**Explanation:**

* **!DOCTYPE beginnersbook** defines that this is the beginning of the DTD declaration and the root element of this XML document is beginnersbook
* **!ELEMENT beginnersbook** defines that the root element beginnersbook must contain four elements: “to,from,subject,message”
* **!ELEMENT to** defines that the element “to” is of type “#PCDATA” where “#PCDATA” stands for Parsed Character Data which means that this data is parsable by XML parser
* **!ELEMENT from** defines that the element “from” is of type “#PCDATA”
* **!ELEMENT subject** defines that the element “subject” is of type “#PCDATA”
* **!ELEMENT message** defines that the element “message” is of type “#PCDATA”

## External DTD Declaration

In the above example, we have **internal DTD declaration**. Lets see how we can have **external DTD declaration** in an XML document. For the understanding purpose, lets take the same above example here –

To have the external DTD declaration in an XML document, we must include the reference to the DTD file in the <!DOCTYPE> definition, as we have done in the following example.

<?xml version="1.0"?>

<!DOCTYPE beginnersbook SYSTEM "bb.dtd">

<beginnersbook>

<to>My Readers</to>

<from>Chaitanya</from>

<subject>A Message to my readers</subject>

<message>Welcome to beginnersbook.com</message>

</beginnersbook>

The <!DOCTYPE> definition in the above document contains the reference to “bb.dtd” file. Here is the content of “bb.dtd” file that contains the DTD for above XML document –

<!ELEMENT beginnersbook (to,from,subject,message)>

<!ELEMENT to (#PCDATA)>

<!ELEMENT from (#PCDATA)>

<!ELEMENT subject (#PCDATA)>

<!ELEMENT message (#PCDATA)>

# XML Schema – XSD (XML Schema Definition)

BY CHAITANYA SINGH | FILED UNDER: [**XML TUTORIAL**](https://beginnersbook.com/category/xml-tutorial/)

we learned about XML DTD which is used for XML validation. Similar to DTD, XML Schema is also used to check whether the given XML document is “well formed” and “valid”. In this guide, we will learn what is an **XML Schema**, how it is better than DTD and Schema examples.

## XML XSD (XML Schema Definition)

XML schema is an alternative to DTD. An XML document is considered “well formed” and “valid” if it is successfully validated against XML Schema. The extension of Schema file is .xsd.

## An example of XML Schema

**XML file: bb.xml**

<?xml version="1.0"?>

<beginnersbook>

<to>My Readers</to>

<from>Chaitanya</from>

<subject>A Message to my readers</subject>

<message>Welcome to beginnersbook.com</message>

</beginnersbook>

**XML Schema file: bb.xsd**

<?xml version="1.0"?>

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"

targetNamespace="https://www.beginnersbook.com"

xmlns="https://www.beginnersbook.com"

elementFormDefault="qualified">

<xs:element name="beginnersbook">

  <xs:complexType>

    <xs:sequence>

      <xs:element name="to" type="xs:string"/>

      <xs:element name="from" type="xs:string"/>

      <xs:element name="subject" type="xs:string"/>

      <xs:element name="message" type="xs:string"/>

    </xs:sequence>

  </xs:complexType>

</xs:element>

</xs:schema>

**Explanation:**

* <xs:element name=”beginnersbook”> defines that beginnersbook is the name of an element.
* <xs:complexType> This is the next line after the element “beginnersbook”. It defines the type of element “beginnersbook”, it says that the type of this element is “complexType” (we will discuss this type later in this same tutorial)
* <xs:sequence> It defines that the complex type element “beginnersbook” is a sequence of elements
* <xs:element name=”to” type=”xs:string”> It defines that the element “to” is of type string
* <xs:element name=”from” type=”xs:string”> It defines that the element “from” is of type string
* <xs:element name=”subject” type=”xs:string”> It defines that the element “subject” is of type string
* <xs:element name=”message” type=”xs:string”> It defines that the element “message” is of type string

## XML Schema Data types

In the above example, we have seen that the root element “beginnersbook” is complexType. In XML schema an element belongs to either of the following two types.

1. simpleType – A singleType element can contain text, they do not contain other elements. In the above example, the elements to, from, subject and message are simpleType element.

2. complexType – A complexType element can contain attributes, other elements, and text. In the above example, the element beginnersbook is of type complexType because it contains other elements.

## Advantages of using XML Schema over DTD

1. Schema uses XML as language so you don’t have to learn new syntax.  
2. XML schema supports data types and namespaces.  
3. You can use XML parser to parse the XML schema as well.  
4. Just like XML, the XML schema is extensible which means you can reuse the schema in other schema, as well as you can reference more than one schemas in a single XML document.