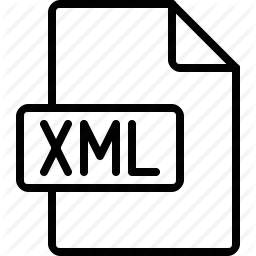
XML Tutorial



XML tutorial is designed for beginners and professionals. Our XML tutorial provides a detailed knowledge of XML technology like what is xml, features of xml, xml example, xml related technologies, creating xml structure by DTD, creating xml structure by schema (XSD), difference between DTD and schema.

XML Validation

XML file can be validated by 2 ways:

1. against DTD
2. against XSD

DTD (Document Type Definition) and XSD (XML Schema Definition) are used to define XML structure.

XML DTD

In our XML tutorial, you will learn about DTD file, creating xml with DTD, using CSS file, CDATA vs PCDATA and difference between DTD and XML schema.

Let's see an example of XML using DTD file.

*employee.xml*

SQL CREATE TABLE

1. **<?xml** version="1.0"**?>**
2. <!DOCTYPE employee SYSTEM "employee.dtd"**>**
3. **<employee>**
4. **<firstname>**vimal**</firstname>**
5. **<lastname>**jaiswal**</lastname>**
6. **<email>**vimal@javatpoint.com**</email>**
7. **</employee>**

[**Test it Now**](http://www.javatpoint.com/xmlpages/pcdata.xml)

A detailed description of XML with DTD is given in next pages.

XML Schema

In this XML tutorial, we will provide a detail description of schema file, XML schema validation, XML schema data types and XML parsers.

Let's see an example of XML using XSD file.

1. **<?xml** version="1.0"**?>**
2. **<employee**
3. xmlns="http://www.javatpoint.com"
4. xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5. xsi:schemaLocation="http://www.javatpoint.com employee.xsd"**>**
6. **<firstname>**vimal**</firstname>**
7. **<lastname>**jaiswal**</lastname>**
8. **<email>**vimal@javatpoint.com**</email>**
9. **</employee>**

[**Test it Now**](http://www.javatpoint.com/xmlpages/xmlschema1.xml)

A detailed description of XML with XSD is given in next pages.

# What is xml

* **Xml** (eXtensible Markup Language) is a mark up language.
* XML is designed to store and transport data.
* Xml was released in late 90’s. it was created to provide an easy to use and store self describing data.
* XML became a W3C Recommendation on February 10, 1998.
* XML is not a replacement for HTML.
* XML is designed to be self-descriptive.
* XML is designed to carry data, not to display data.
* XML tags are not predefined. You must define your own tags.
* XML is platform independent and language independent.

#### **Note: Self-describing data is the data that describes both its content and structure.**

## What is mark-up language

A **mark up language** is a modern system for highlight or underline a document.

Students often underline or highlight a passage to revise easily, same in the sense of modern mark up language highlighting or underlining is replaced by tags.

## Prerequisite

Before you start to learn xml, you should know basic of HTML & JavaScript.

## Why xml

**Platform Independent and Language Independent:** The main benefit of xml is that you can use it to take data from a program like Microsoft SQL, convert it into XML then share that XML with other programs and platforms. You can communicate between two platforms which are generally very difficult.

The main thing which makes XML truly powerful is its international acceptance. Many corporation use XML interfaces for databases, programming, office application mobile phones and more. It is due to its platform independent feature.

Next Topic[XML Features and Advantages](https://www.javatpoint.com/features-and-advantages-of-xml)

# Features and Advantages of XML

XML is widely used in the era of web development. It is also used to simplify data storage and data sharing.

The main features or advantages of XML are given below.

## 1) 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 focus 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.

## 2) XML simplifies data sharing

In the real world, computer systems and databases contain data in incompatible formats.

Exception Handling in Java - Javatpoint

XML data is stored in plain text format. This provides a software- and hardware-independent way of storing data.

This makes it much easier to create data that can be shared by different applications.

## 3) 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.

#### **4) XML simplifies Platform change**

Upgrading to new systems (hardware or software platforms), 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.

## 5) XML increases data availability

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), and make it more available for blind people, or people with other disabilities.

## 6) XML can be used to create new internet languages

A lot of new Internet languages are created with XML.

Here are some examples:

* **XHTML**
* **WSDL** for describing available web services
* **WAP** and **WML** as markup languages for handheld devices
* **RSS** languages for news feeds
* **RDF** and **OWL** for describing resources and ontology
* **SMIL** for describing multimedia for the web

Next Topic[XML Example](https://www.javatpoint.com/xml-example)

XML Example

XML documents create a hierarchical structure looks like a tree so it is known as XML Tree that starts at "the root" and branches to "the leaves".

Example of XML Document

XML documents uses a self-describing and simple syntax:

1. **<?xml** version="1.0" encoding="ISO-8859-1"**?>**
2. **<note>**
3. **<to>**Tove**</to>**
4. **<from>**Jani**</from>**
5. **<heading>**Reminder**</heading>**
6. **<body>**Don't forget me this weekend!**</body>**
7. **</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"):

1. **<note>**

The next 4 lines describe 4 child elements of the root (to, from, heading, and body).

1. **<to>**Tove**</to>**
2. **<from>**Jani**</from>**
3. **<heading>**Reminder**</heading>**
4. **<body>**Don't forget me this weekend!**</body>**

And finally the last line defines the end of the root element.

Exception Handling in Java - Javatpoint

1. **</note>**

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).

1. **<root>**
2. **<child>**
3. **<subchild>**.....**</subchild>**
4. **</child>**
5. **</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).

Another Example of XML: Books

*File: books.xml*

1. **<bookstore>**
2. **<book** category="COOKING"**>**
3. **<title** lang="en"**>**Everyday Italian**</title>**
4. **<author>**Giada De Laurentiis**</author>**
5. **<year>**2005**</year>**
6. **<price>**30.00**</price>**
7. **</book>**
8. **<book** category="CHILDREN"**>**
9. **<title** lang="en"**>**Harry Potter**</title>**
10. **<author>**J K. Rowling**</author>**
11. **<year>**2005**</year>**
12. **<price>**29.99**</price>**
13. **</book>**
14. **<book** category="WEB"**>**
15. **<title** lang="en"**>**Learning XML**</title>**
16. **<author>**Erik T. Ray**</author>**
17. **<year>**2003**</year>**
18. **<price>**39.95**</price>**
19. **</book>**
20. **</bookstore>**

[**Test it Now**](http://www.javatpoint.com/xmlpages/books.xml)

The root element in the example is <bookstore>. All elements in the document are contained within <bookstore>.

The <book> element has 4 children: <title>,< author>, <year> and <price>.

Another Example of XML: Emails

*File: emails.xml*

1. **<?xml** version="1.0" encoding="UTF-8"**?>**
2. **<emails>**
3. **<email>**
4. **<to>**Vimal**</to>**
5. **<from>**Sonoo**</from>**
6. **<heading>**Hello**</heading>**
7. **<body>**Hello brother, how are you!**</body>**
8. **</email>**
9. **<email>**
10. **<to>**Peter**</to>**
11. **<from>**Jack**</from>**
12. **<heading>**Birth day wish**</heading>**
13. **<body>**Happy birth day Tom!**</body>**
14. **</email>**
15. **<email>**
16. **<to>**James**</to>**
17. **<from>**Jaclin**</from>**
18. **<heading>**Morning walk**</heading>**
19. **<body>**Please start morning walk to stay fit!**</body>**
20. **</email>**
21. **<email>**
22. **<to>**Kartik**</to>**
23. **<from>**Kumar**</from>**
24. **<heading>**Health Tips**</heading>**
25. **<body>**Smoking is injurious to health!**</body>**
26. **</email>**
27. **</emails>**

[**Test it Now**](http://www.javatpoint.com/xmlpages/emails.xml)

Next Topic[XML Related Technologies](https://www.javatpoint.com/xml-related-technologies)

# XML Related Technologies

Here we have pointed out XML related technologies. There are following XML related technologies:

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Technology** | **Meaning** | **Description** |
| 1) | XHTML | Extensible html | It is a clearer and stricter version of XML. It belongs to the family of XML markup languages. It was developed to make html more extensible and increase inter-operability with other data. |
| 2) | XML DOM | XML document object model | It is a standard document model that is used to access and manipulate XML. It defines the XML file in tree structure. |
| 3) | XSL it contain three parts: i) XSLT (xsl transform) ii) XSL iii)XPath | Extensible style sheet language | i) It transforms XML into other formats, like html. ii) It is used for formatting XML to screen, paper etc. iii) It is a language to navigate XML documents. |
| 4) | XQuery | XML query language | It is a XML based language which is used to query XML based data. |
| 5) | DTD | Document type definition | It is an standard which is used to define the legal elements in an XML document. |
| 6) | XSD | XML schema definition | It is an XML based alternative to dtd. It is used to describe the structure of an XML document. |
| 7) | XLink | XML linking language | xlink stands for XML linking language. This is a language for creating hyperlinks (external and internal links) in XML documents. |
| 8) | XPointer | XML pointer language | It is a system for addressing components of XML based internet media. It allows the xlink hyperlinks to point to more specific parts in the XML document. |
| 9) | SOAP | Simple object access protocol | It is an acronym stands simple object access protocol. It is XML based protocol to let applications exchange information over http. in simple words you can say that it is protocol used for accessing web services. |
| 10) | WSDL | web services description languages | It is an XML based language to describe web services. It also describes the functionality offered by a web service. |
| 11) | RDF | Resource description framework | RDF is an XML based language to describe web resources. It is a standard model for data interchange on the web. It is used to describe the title, author, content and copyright information of a web page. |
| 12) | SVG | Scalable vector graphics | It is an XML based vector image format for two-dimensional images. It defines graphics in XML format. It also supports animation. |
| 13) | RSS | Really simple syndication | RSS is a XML-based format to handle web content syndication. It is used for fast browsing for news and updates. It is generally used for news like sites. |

# XML Attributes

XML elements can have attributes. By the use of attributes we can add the information about the element.

XML attributes enhance the properties of the elements.

#### **Note: XML attributes must always be quoted. We can use single or double quote.**

Let us take an example of a book publisher. Here, book is the element and publisher is the attribute.

1. **<book** publisher="Tata McGraw Hill"**></book>**

Or

1. **<book** publisher='Tata McGraw Hill'**></book>**

**Metadata should be stored as attribute and data should be stored as element.**

1. **<book>**
2. **<book** category="computer"**>**
3. **<author>** A & B **</author>**
4. **</book>**

Data can be stored in attributes or in child elements. But there are some limitations in using attributes, over child elements.

SQL CREATE TABLE

## Why should we avoid XML attributes

* Attributes cannot contain multiple values but child elements can have multiple values.
* Attributes cannot contain tree structure but child element can.
* Attributes are not easily expandable. If you want to change in attribute's vales in future, it may be complicated.
* Attributes cannot describe structure but child elements can.
* Attributes are more difficult to be manipulated by program code.
* Attributes values are not easy to test against a DTD, which is used to define the legal elements of an XML document.

## Difference between attribute and sub-element

In the context of documents, attributes are part of markup, while sub elements are part of the basic document contents.

In the context of data representation, the difference is unclear and may be confusing.

Same information can be represented in two ways:

**1st way:**

1. **<book** publisher="Tata McGraw Hill"**>** **</book>**

**2nd way:**

1. **<book>**
2. **<publisher>** Tata McGraw Hill **</publisher>**
3. **</book>**

In the first example publisher is used as an attribute and in the second example publisher is an element.

Both examples provide the same information but it is good practice to avoid attribute in XML and use elements instead of attributes.

# XML Comments

XML comments are just like HTML comments. We know that the comments are used to make codes more understandable other developers.

XML Comments add notes or lines for understanding the purpose of an XML code. Although XML is known as self-describing data but sometimes XML comments are necessary.

#### **Syntax**

An XML comment should be written as:

1. <!-- Write your comment-->

#### **You cannot nest one XML comment inside the another.**

## XML Comments Example

Let's take an example to show the use of comment in an XML example:

1. **<?xml** version="1.0" encoding="UTF-8" **?>**
2. <!--Students marks are uploaded by months-->
3. **<students>**
4. **<student>**
5. **<name>**Ratan**</name>**
6. **<marks>**70**</marks>**
7. **</student>**
8. **<student>**
9. **<name>**Aryan**</name>**
10. **<marks>**60**</marks>**
11. **</student>**
12. **</students>**

[**Test it Now**](http://www.javatpoint.com/xmlpages/xmlcomments1.xml)

## Rules for adding XML comments

* Don't use a comment before an XML declaration.
* You can use a comment anywhere in XML document except within attribute value.
* Don't nest a comment inside the other comment.

Next Topic[XML Tree](https://www.javatpoint.com/xml-tree-structure)

# XML Tree Structure

An XML document has a self descriptive structure. It forms a tree structure which is referred as an XML tree. The tree structure makes easy to describe an XML document.

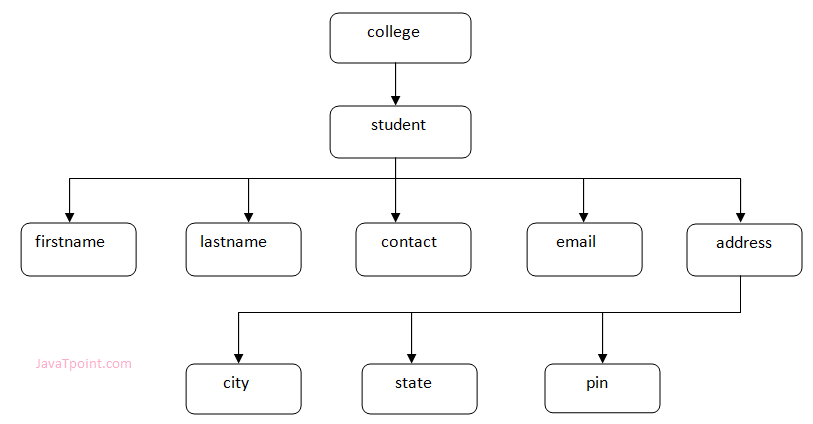
A tree structure contains root element (as parent), child element and so on. It is very easy to traverse all succeeding branches and sub-branches and leaf nodes starting from the root.

## Example of an XML document

1. **<?xml** version="1.0"**?>**
2. **<college>**
3. **<student>**
4. **<firstname>**Tamanna**</firstname>**
5. **<lastname>**Bhatia**</lastname>**
6. **<contact>**09990449935**</contact>**
7. **<email>**tammanabhatia@abc.com**</email>**
8. **<address>**
9. **<city>**Ghaziabad**</city>**
10. **<state>**Uttar Pradesh**</state>**
11. **<pin>**201007**</pin>**
12. **</address>**
13. **</student>**
14. **</college>**

[**Test it Now**](http://www.javatpoint.com/xmlpages/xmltree.xml)

Let's see the tree-structure representation of the above example.



In the above example, first line is the XML declaration. It defines the XML version 1.0. Next line shows the root element (college) of the document. Inside that there is one more element (student). Student element contains five branches named <firstname>, <lastname>, <contact>, <Email> and <address>.

<address> branch contains 3 sub-branches named <city>, <state> and <pin>.

Triggers in SQL (Hindi)

#### **Note: DOM parser represents the XML document in Tree structure.**

## XML Tree Rules

These rules are used to figure out the relationship of the elements. It shows if an element is a child or a parent of the other element.

**Descendants:** If element A is contained by element B, then A is known as descendant of B. In the above example "College" is the root element and all the other elements are the descendants of "College".

**Ancestors:** The containing element which contains other elements is called "Ancestor" of other element. In the above example Root element (College) is ancestor of all other elements.

Next Topic[XML Validation](https://www.javatpoint.com/xml-validation)

XML Validation

A well formed XML document can be validated against DTD or Schema.

A well-formed XML document is an XML document with correct syntax. It is very necessary to know about valid XML document before knowing XML validation.

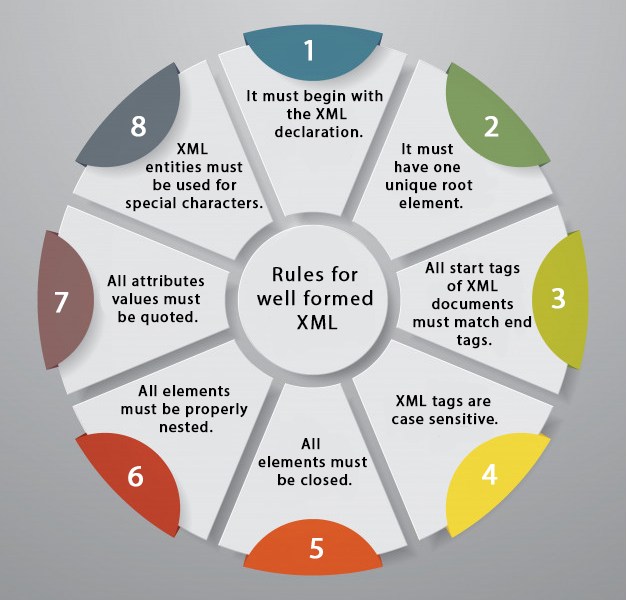
Valid XML document

It must be well formed (satisfy all the basic syntax condition)

It should be behave according to predefined DTD or XML schema

Rules for well formed XML

* It must begin with the XML declaration.
* It must have one unique root element.
* All start tags of XML documents must match end tags.
* XML tags are case sensitive.
* All elements must be closed.
* All elements must be properly nested.
* All attributes values must be quoted.
* XML entities must be used for special characters.



XML DTD

A DTD defines the legal elements of an XML document

In simple words we can say that a DTD defines the document structure with a list of legal elements and attributes.

XML schema is a XML based alternative to DTD.

Actually DTD and XML schema both are used to form a well formed XML document.

We should avoid errors in XML documents because they will stop the XML programs.

XML schema

It is defined as an XML language

Uses namespaces to allow for reuses of existing definitions

It supports a large number of built in data types and definition of derived data types

Next Topic[XML DTD](https://www.javatpoint.com/xml-dtd)

# XML DTD

## What is DTD

DTD stands for **Document Type Definition**. It defines the legal building blocks of an XML document. It is used to define document structure with a list of legal elements and attributes.

## Purpose of DTD

Its main purpose is to define the structure of an XML document. It contains a list of legal elements and define the structure with the help of them.

## Checking Validation

Before proceeding with XML DTD, you must check the validation. An XML document is called "well-formed" if it contains the correct syntax.

A well-formed and valid XML document is one which have been validated against DTD.

Visit **http://www.xmlvalidation.com** to validate the XML file.

## Valid and well-formed XML document with DTD

Let's take an example of well-formed and valid XML document. It follows all the rules of DTD.

Difference between JDK, JRE, and JVM

*employee.xml*

1. **<?xml** version="1.0"**?>**
2. <!DOCTYPE employee SYSTEM "employee.dtd"**>**
3. **<employee>**
4. **<firstname>**vimal**</firstname>**
5. **<lastname>**jaiswal**</lastname>**
6. **<email>**vimal@javatpoint.com**</email>**
7. **</employee>**

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

*employee.dtd*

1. <!ELEMENT employee (firstname,lastname,email)**>**
2. <!ELEMENT firstname (#PCDATA)**>**
3. <!ELEMENT lastname (#PCDATA)**>**
4. <!ELEMENT email (#PCDATA)**>**

[**Test it Now**](http://www.javatpoint.com/xmlpages/pcdata.xml)

## Description of DTD

**<!DOCTYPE employee :** It defines that the root element of the document is employee.

**<!ELEMENT employee:** It defines that the employee element contains 3 elements "firstname, lastname and email".

**<!ELEMENT firstname:** It defines that the firstname element is #PCDATA typed. (parse-able data type).

**<!ELEMENT lastname:** It defines that the lastname element is #PCDATA typed. (parse-able data type).

**<!ELEMENT email:** It defines that the email element is #PCDATA typed. (parse-able data type).

## XML DTD with entity declaration

A doctype declaration can also define special strings that can be used in the XML file.

An entity has three parts:

1. An ampersand (&)
2. An entity name
3. A semicolon (;)

Syntax to declare entity:

1. <!ENTITY entity-name "entity-value"**>**

Let's see a code to define the ENTITY in doctype declaration.

author.xml

1. **<?xml** version="1.0" standalone="yes" **?>**
2. <!DOCTYPE author [
3. <!ELEMENT author (#PCDATA)**>**
4. <!ENTITY sj "Sonoo Jaiswal"**>**
5. ]**>**
6. **<author>**&sj;**</author>**

[**Test it Now**](http://www.javatpoint.com/xmlpages/author.xml)

In the above example, sj is an entity that is used inside the author element. In such case, it will print the value of sj entity that is "Sonoo Jaiswal".

#### **Note: A single DTD can be used in many XML files.**

# XML CSS

## Purpose of CSS in XML

CSS (Cascading Style Sheets) can be used to add style and display information to an XML document. It can format the whole XML document.

## How to link XML file with CSS

To link XML files with CSS, you should use the following syntax:

1. **<?xml-stylesheet** type="text/css" href="cssemployee.css"**?>**

## XML CSS Example

Let's see the css file.

cssemployee.css

1. employee
2. {
3. background-color: pink;
4. }
5. firstname,lastname,email
6. {
7. font-size:25px;
8. display:block;
9. color: blue;
10. margin-left: 50px;
11. }

Let's create the DTD file.

*employee.dtd*

1. <!ELEMENT employee (firstname,lastname,email)**>**
2. <!ELEMENT firstname (#PCDATA)**>**
3. <!ELEMENT lastname (#PCDATA)**>**
4. <!ELEMENT email (#PCDATA)**>**

Let's see the xml file using CSS and DTD.

*employee.xml*

1. **<?xml** version="1.0"**?>**
2. **<?xml-stylesheet** type="text/css" href="cssemployee.css"**?>**
3. <!DOCTYPE employee SYSTEM "employee.dtd"**>**
4. **<employee>**
5. **<firstname>**vimal**</firstname>**
6. **<lastname>**jaiswal**</lastname>**
7. **<email>**vimal@javatpoint.com**</email>**
8. **</employee>**

[**Test it Now**](http://www.javatpoint.com/xmlpages/xmlcss.xml)

#### **CSS is not generally used to format XML file. W3C recommends XSLT instead of CSS.**

XML Schema

What is XML schema

XML schema is a language which is used for expressing constraint about XML documents. There are so many schema languages which are used now a days for example Relax- NG and XSD (XML schema definition).

An XML schema is used to define the structure of an XML document. It is like DTD but provides more control on XML structure.

Checking Validation

An XML document is called "well-formed" if it contains the correct syntax. A well-formed and valid XML document is one which have been validated against Schema.

Visit **http://www.xmlvalidation.com** to validate the XML file against schema or DTD.

XML Schema Example

Let's create a schema file.

3.4M

81

How to find Nth Highest Salary in SQL

*employee.xsd*

1. **<?xml** version="1.0"**?>**
2. **<xs:schema** xmlns:xs="http://www.w3.org/2001/XMLSchema"
3. targetNamespace="http://www.javatpoint.com"
4. xmlns="http://www.javatpoint.com"
5. elementFormDefault="qualified"**>**
7. **<xs:element** name="employee"**>**
8. **<xs:complexType>**
9. **<xs:sequence>**
10. **<xs:element** name="firstname" type="xs:string"**/>**
11. **<xs:element** name="lastname" type="xs:string"**/>**
12. **<xs:element** name="email" type="xs:string"**/>**
13. **</xs:sequence>**
14. **</xs:complexType>**
15. **</xs:element>**
17. **</xs:schema>**

Let's see the xml file using XML schema or XSD file.

*employee.xml*

1. **<?xml** version="1.0"**?>**
2. **<employee**
3. xmlns="http://www.javatpoint.com"
4. xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5. xsi:schemaLocation="http://www.javatpoint.com employee.xsd"**>**
7. **<firstname>**vimal**</firstname>**
8. **<lastname>**jaiswal**</lastname>**
9. **<email>**vimal@javatpoint.com**</email>**
10. **</employee>**

[**Test it Now**](http://www.javatpoint.com/xmlpages/xmlschema.xml)

Description of XML Schema

**<xs:element name="employee">** : It defines the element name employee.

**<xs:complexType>** : It defines that the element 'employee' is complex type.

**<xs:sequence>** : It defines that the complex type is a sequence of elements.

**<xs:element name="firstname" type="xs:string"/>** : It defines that the element 'firstname' is of string/text type.

**<xs:element name="lastname" type="xs:string"/>** : It defines that the element 'lastname' is of string/text type.

**<xs:element name="email" type="xs:string"/>** : It defines that the element 'email' is of string/text type.

XML Schema Data types

There are two types of data types in XML schema.

1. simpleType
2. complexType

simpleType

The simpleType allows you to have text-based elements. It contains less attributes, child elements, and cannot be left empty.

complexType

The complexType allows you to hold multiple attributes and elements. It can contain additional sub elements and can be left empty.

Next Topic[DTD vs XSD](https://www.javatpoint.com/dtd-vs-xsd)

# DTD vs XSD

There are many differences between DTD (Document Type Definition) and XSD (XML Schema Definition). In short, DTD provides less control on XML structure whereas XSD (XML schema) provides more control.

The important differences are given below:

|  |  |  |
| --- | --- | --- |
| **No.** | **DTD** | **XSD** |
| 1) | DTD stands for **Document Type Definition**. | XSD stands for XML Schema Definition. |
| 2) | DTDs are derived from **SGML** syntax. | XSDs are written in XML. |
| 3) | DTD **doesn't support datatypes**. | XSD **supports datatypes** for elements and attributes. |
| 4) | DTD **doesn't support namespace**. | XSD **supports namespace**. |
| 5) | DTD **doesn't define order** for child elements. | XSD **defines order** for child elements. |
| 6) | DTD is **not extensible**. | XSD is **extensible**. |
| 7) | DTD is **not simple to learn**. | XSD is **simple to learn** because you don't need to learn new language. |
| 8) | DTD provides **less control** on XML structure. | XSD provides **more control** on XML structure. |

Next Topic[CDATA vs PCDATA](https://www.javatpoint.com/cdata-pcdata)

[← Prev](https://www.javatpoint.com/xml-schema)[Next →](https://www.javatpoint.com/cdata-pcdata)

CDATA vs PCDATA

CDATA

CDATA: (Unparsed Character data): CDATA contains the text which is not parsed further in an XML document. Tags inside the CDATA text are not treated as markup and entities will not be expanded.

Let's take an example for CDATA:

1. **<?xml** version="1.0"**?>**
2. <!DOCTYPE employee SYSTEM "employee.dtd"**>**
3. **<employee>**
4. <![CDATA[
5. <firstname>vimal</firstname>
6. <lastname>jaiswal</lastname>
7. <email>vimal@javatpoint.com</email>
8. ]]>
9. **</employee>**

[**Test it Now**](http://www.javatpoint.com/xmlpages/cdata.xml)

In the above CDATA example, CDATA is used just after the element employee to make the data/text unparsed, so it will give the value of employee:

<firstname>vimal</firstname><lastname>jaiswal</lastname><email>vimal@javatpoint.com</email>

PCDATA

PCDATA: (Parsed Character Data): XML parsers are used to parse all the text in an XML document. PCDATA stands for Parsed Character data. PCDATA is the text that will be parsed by a parser. Tags inside the PCDATA will be treated as markup and entities will be expanded.

In other words you can say that a parsed character data means the XML parser examine the data and ensure that it doesn't content entity if it contains that will be replaced.

Let's take an example:

Difference between JDK, JRE, and JVM

1. **<?xml** version="1.0"**?>**
2. <!DOCTYPE employee SYSTEM "employee.dtd"**>**
3. **<employee>**
4. **<firstname>**vimal**</firstname>**
5. **<lastname>**jaiswal**</lastname>**
6. **<email>**vimal@javatpoint.com**</email>**
7. **</employee>**

[**Test it Now**](http://www.javatpoint.com/xmlpages/pcdata.xml)

In the above example, the employee element contains 3 more elements 'firstname', 'lastname', and 'email', so it parses further to get the data/text of firstname, lastname and email to give the value of employee as:

vimal jaiswal vimal@javatpoint.com

Next Topic[XML Parsers](https://www.javatpoint.com/xml-parsers)

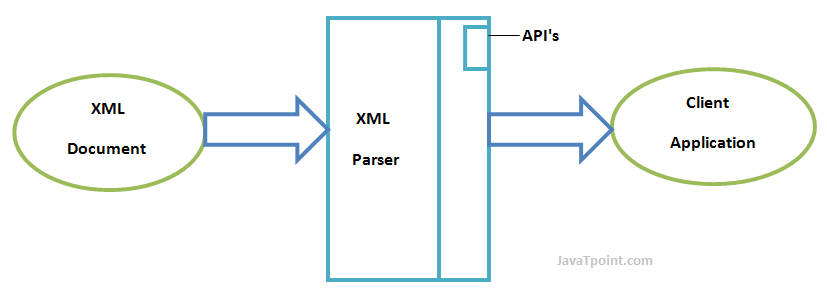
[← Prev](https://www.javatpoint.com/dtd-vs-xsd)[Next →](https://www.javatpoint.com/xml-parsers)

# XML Parsers

An XML parser is a software library or package that provides interfaces for client applications to work with an XML document. The XML Parser is designed to read the XML and create a way for programs to use XML.

XML parser validates the document and check that the document is well formatted.

Let's understand the working of XML parser by the figure given below:



## Types of XML Parsers

These are the two main types of XML Parsers:

1. DOM
2. SAX

## DOM (Document Object Model)

A DOM document is an object which contains all the information of an XML document. It is composed like a tree structure. The DOM Parser implements a DOM API. This API is very simple to use.

### **Features of DOM Parser**

A DOM Parser creates an internal structure in memory which is a DOM document object and the client applications get information of the original XML document by invoking methods on this document object.

DOM Parser has a tree based structure.

### **Advantages**

1) It supports both read and write operations and the API is very simple to use.

2) It is preferred when random access to widely separated parts of a document is required.

### **Disadvantages**

1) It is memory inefficient. (consumes more memory because the whole XML document needs to loaded into memory).

2) It is comparatively slower than other parsers.

## SAX (Simple API for XML)

A SAX Parser implements SAX API. This API is an event based API and less intuitive.

### **Features of SAX Parser**

It does not create any internal structure.

Clients does not know what methods to call, they just overrides the methods of the API and place his own code inside method.

It is an event based parser, it works like an event handler in Java.

### **Advantages**

1) It is simple and memory efficient.

2) It is very fast and works for huge documents.

### **Disadvantages**

1) It is event-based so its API is less intuitive.

2) Clients never know the full information because the data is broken into pieces.

Next Topic[XML DOM](https://www.javatpoint.com/xml-dom)

[← Prev](https://www.javatpoint.com/cdata-pcdata)[Next →](https://www.javatpoint.com/xml-dom)

XML DOM

What is XML DOM

DOM is an acronym stands for Document Object Model. It defines a standard way to access and manipulate documents. The Document Object Model (DOM) is a programming API for HTML and XML documents. It defines the logical structure of documents and the way a document is accessed and manipulated.

As a W3C specification, one important objective for the Document Object Model is to provide a standard programming interface that can be used in a wide variety of environments and applications. The Document Object Model can be used with any programming language.

XML DOM defines a standard way to access and manipulate XML documents.

What does XML DOM

The XML DOM makes a tree-structure view for an XML document.

We can access all elements through the DOM tree.

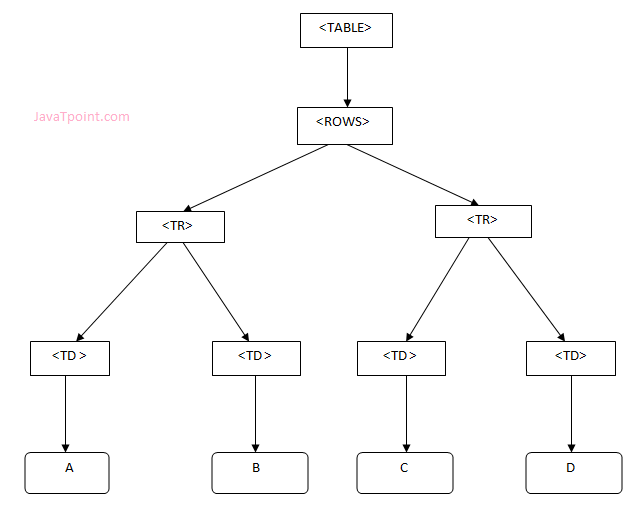
We can modify or delete their content and also create new elements. The elements, their content (text and attributes) are all known as nodes.

Exception Handling in Java - Javatpoint

For example, consider this table, taken from an HTML document:

1. **<TABLE>**
2. **<ROWS>**
3. **<TR>**
4. **<TD>**A**</TD>**
5. **<TD>**B**</TD>**
6. **</TR>**
7. **<TR>**
8. **<TD>**C**</TD>**
9. **<TD>**D**</TD>**
10. **</TR>**
11. **</ROWS>**
12. **</TABLE>**

The Document Object Model represents this table like this:



XML DOM Example : Load XML File

Let's take an example to show how an XML document ("note.xml") is parsed into an XML DOM object.

This example parses an XML document (note.xml) into an XML DOM object and extracts information from it with JavaScript.

Let's see the XML file that contains message.

*note.xml*

1. **<?xml** version="1.0" encoding="ISO-8859-1"**?>**
2. **<note>**
3. **<to>**sonoojaiswal@javatpoint.com**</to>**
4. **<from>**vimal@javatpoint.com**</from>**
5. **<body>**Hello XML DOM**</body>**
6. **</note>**

Let's see the HTML file that extracts the data of XML document using DOM.

*xmldom.html*

1. <!DOCTYPE html**>**
2. **<html>**
3. **<body>**
4. **<h1>**Important Note**</h1>**
5. **<div>**
6. **<b>**To:**</b>** **<span** id="to"**></span><br>**
7. **<b>**From:**</b>** **<span** id="from"**></span><br>**
8. **<b>**Message:**</b>** **<span** id="message"**></span>**
9. **</div>**
10. **<script>**
11. if (window.XMLHttpRequest)
12. {// code for IE7+, Firefox, Chrome, Opera, Safari
13. xmlhttp=new XMLHttpRequest();
14. }
15. else
16. {// code for IE6, IE5
17. xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
18. }
19. xmlhttp.open("GET","note.xml",false);
20. xmlhttp.send();
21. xmlDoc=xmlhttp.responseXML;
22. document.getElementById("to").innerHTML=
23. xmlDoc.getElementsByTagName("to")[0].childNodes[0].nodeValue;
24. document.getElementById("from").innerHTML=
25. xmlDoc.getElementsByTagName("from")[0].childNodes[0].nodeValue;
26. document.getElementById("message").innerHTML=
27. xmlDoc.getElementsByTagName("body")[0].childNodes[0].nodeValue;
28. **</script>**
29. **</body>**
30. **</html>**

[**Test it Now**](http://www.javatpoint.com/oprweb/test.jsp?filename=xmldom1)

Output:

**Important Note**

**To:** sonoojaiswal@javatpoint.com  
**From:** vimal@javatpoint.com  
**Message:** Hello XML DOM

XML DOM Example : Load XML String

This example parses an XML string into an XM DOM object and then extracts some information from it with a JavaScript.

Let's see the HTML file that extracts the data of XML string using DOM.

*xmldom.html*

1. <!DOCTYPE html**>**
2. **<html>**
3. **<body>**
4. **<h1>**Important Note2**</h1>**
5. **<div>**
6. **<b>**To:**</b>** **<span** id="to"**></span><br>**
7. **<b>**From:**</b>** **<span** id="from"**></span><br>**
8. **<b>**Message:**</b>** **<span** id="message"**></span>**
9. **</div>**
10. **<script>**
11. txt1="<note>";
12. txt2="<to>Sania Mirza</to>";
13. txt3="<from>Serena William</from>";
14. txt4="<body>Don't forget me this weekend!</body>";
15. txt5="</note>";
16. txt=txt1+txt2+txt3+txt4+txt5;
18. if (window.DOMParser)
19. {
20. parser=new DOMParser();
21. xmlDoc=parser.parseFromString(txt,"text/xml");
22. }
23. else // Internet Explorer
24. {
25. xmlDoc=new ActiveXObject("Microsoft.XMLDOM");
26. xmlDoc.async=false;
27. xmlDoc.loadXML(txt);
28. }
29. document.getElementById("to").innerHTML=
30. xmlDoc.getElementsByTagName("to")[0].childNodes[0].nodeValue;
31. document.getElementById("from").innerHTML=
32. xmlDoc.getElementsByTagName("from")[0].childNodes[0].nodeValue;
33. document.getElementById("message").innerHTML=
34. xmlDoc.getElementsByTagName("body")[0].childNodes[0].nodeValue;
35. **</script>**
36. **</body>**
37. **</html>**

[**Test it Now**](http://www.javatpoint.com/oprweb/test.jsp?filename=xmldom2)

Output:

**Important Note2**

**To:** Sania Mirza  
**From:** Serena William  
**Message:** Don't forget me this weekend!

Next Topic[XML Database](https://www.javatpoint.com/xml-database)