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.

**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.

# Features and Advantages of XML

## **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.

## **2) 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 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:

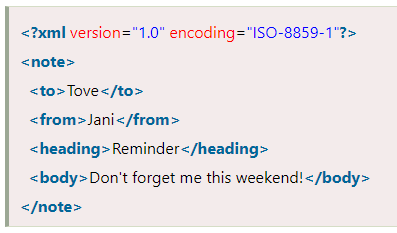
* **XHTML**
* **WSDL** for describing available web services
* **WAP** and **WML** as markup languages for handheld devices
* **RSS** languages for news feeds
* **RDF** and **OWL** for describing resources and ontology
* **SMIL** for describing multimedia for the web

# 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:



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"):

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).

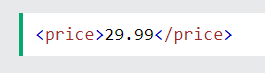


The terms parent, child, and sibling are used to describe the relationships between elements. Parent elements have children. Children on the same level are called siblings (brothers or sisters).

All elements can have text content and attributes (just like in HTML).

## **What is an XML Element?**

An XML element is everything from (including) the element's start tag to (including) the element's end tag.



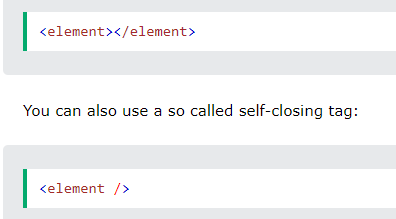
An element can contain:

* text
* attributes
* other elements
* or a mix of the above

## **Empty XML Elements**

An element with no content is said to be empty.

In XML, you can indicate an empty element like this:



## **XML Naming Rules**

XML elements must follow these naming rules:

* Element names are case-sensitive
* Element names must start with a letter or underscore
* Element names cannot start with the letters xml (or XML, or Xml, etc)
* Element names can contain letters, digits, hyphens, underscores, and periods
* Element names cannot contain spaces

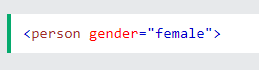
Any name can be used, no words are reserved (except xml).

# XML Attributes

## **XML Attributes Must be Quoted**

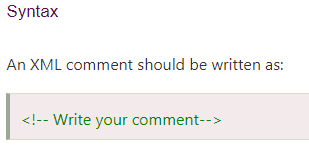
Attribute values must always be quoted. Either single or double quotes can be used.

For a person's gender, the <person> element can be written like this:



# XML Comments

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



# 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

# 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.

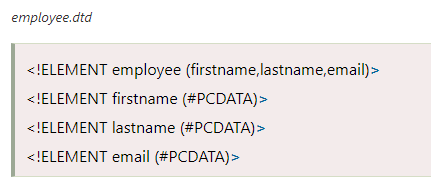
A well-formed and valid XML document is one which have been validated against DTD.

## **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.



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



## **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**

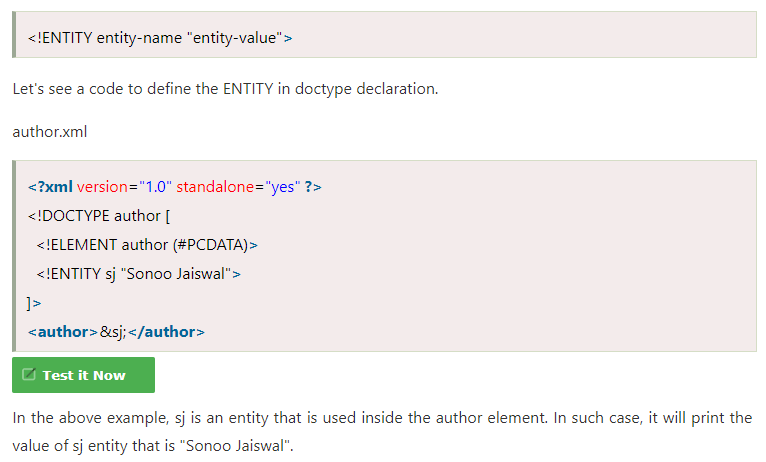
A doctype declaration can also define special strings that can be used in the XML file.

An entity has three parts:

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

**Syntax to declare entity:**

Syntax to declare entity:

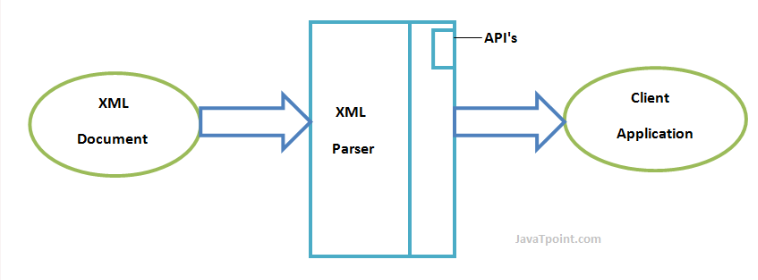


# XML Parsers

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.

# RESTful Web Services

REST stands for REpresentational State Transfer.

REST is an architectural style not a protocol.

## **Advantages of RESTful Web Services**

**Fast**: RESTful Web Services are fast because there is no strict specification like SOAP. It consumes less bandwidth and resource.

**Language and Platform independent**: RESTful web services can be written in any programming language and executed in any platform.

**Can use SOAP**: RESTful web services can use SOAP web services as the implementation.

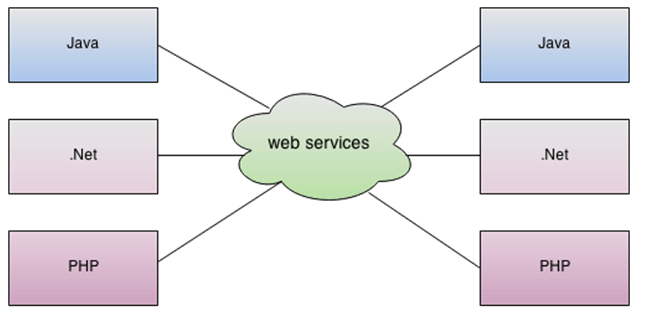
**Permits different data format**: RESTful web service permits different data format such as Plain Text, HTML, XML and JSON.

What is Web Service

A **Web Service** is can be defined by following ways:

* It is a client-server application or application component for communication.
* The method of communication between two devices over the network.
* It is a software system for the interoperable machine to machine communication.
* It is a collection of standards or protocols for exchanging information between two devices or application.

Let's understand it by the figure given below:



As you can see in the figure, Java, .net, and PHP applications can communicate with other applications through web service over the network. For example, the Java application can interact with Java, .Net, and PHP applications. So web service is a language independent way of communication.

## **Types of Web Services**

There are mainly two types of web services.

1. SOAP web services.
2. RESTful web services.

## **Web Service Features**

### **XML-Based**

Web services use XML at data description and data transportation layers. Using XML exclude any networking, operating system, or platform binding. Web services-based operation is extremely interoperable at their core level.

### **Loosely Coupled**

A client of a web service is not fixed to the web service directly. The web service interface can support innovation over time without negotiating the client's ability to communicate with the service. A tightly coupled system means that the client and server logic are closely tied to one another, indicating that if one interface changes, then another must be updated. Accepting a loosely coupled architecture tends to make software systems more manageable and allows more straightforward integration between various systems.

### **Coarse-Grained**

Object-oriented technologies such as Java expose their functions through individual methods. A specific process is too fine an operation to provide any suitable capability at a corporate level. Building a Java program from scratch needed the creation of various fine-grained functions that are then collected into a coarse-grained role that is consumed by either a client or another service.

Businesses and the interfaces that they prove should be coarse-grained. Web services technology implement a natural method of defining coarse-grained services that approach the right amount of business logic.

### **Ability to be Synchronous or Asynchronous**

Synchronicity specifies the binding of the client to the execution of the function. In synchronous invocations, the client blocks and delays in completing its service before continuing. Asynchronous operations grant a client to invoke a task and then execute other functions.

Asynchronous clients fetch their result at a later point in time, while synchronous clients receive their effect when the service has completed. Asynchronous capability is an essential method in enabling loosely coupled systems.

### **Supports Remote Procedure Calls (RPCs)**

Web services allow consumers to invoke procedures, functions, and methods on remote objects using an XML-based protocol. Remote systems expose input and output framework that a web service must support.

Component development through Enterprise JavaBeans (EJBs) and .NET Components has more become a part of architectures and enterprise deployments over a previous couple of years. Both technologies are assigned and accessible through a variety of RPC mechanisms.