**What is XML?**

Extensible Markup Language (XML) lets you define and store data in a shareable manner. XML supports information exchange between computer systems such as websites, databases, and third-party applications. Predefined rules make it easy to transmit data as XML files over any network because the recipient can use those rules to read the data accurately and efficiently.

**Why is XML important?**

Extensible Markup Language (XML) is a markup language that provides rules to define any data. Unlike other programming languages, XML cannot perform computing operations by itself. Instead, any programming language or software can be implemented for structured data management.

For example, consider a text document with comments on it. The comments might give suggestions like these:

* Make the title bold
* This sentence is a header
* This word is the author

Such comments improve the document’s usability without affecting its content. Similarly, XML uses markup symbols to provide more information about any data. Other software, like browsers and data processing applications, use this information to process structured data more efficiently.

**XML tags**

You use markup symbols, called tags in XML, to define data. For example, to represent data for a bookstore, you can create tags such as <book>, <title>, and <author>. Your XML document for a single book would have content like this:

<book>

<title> Learning Amazon Web Services </title>

<author> Mark Wilkins </author>

</book>

Tags bring sophisticated data coding to integrate information flows across different systems.

**What are the benefits of using XML?**

Support interbusiness transactions.

When a company sells a good or service to another company, the two businesses need to exchange information like cost, specifications, and delivery schedules. With Extensible Markup Language (XML), they can share all the necessary information electronically and close complex deals automatically, without any human intervention.

Maintain data integrity.

XML lets you transfer data along with the data’s description, preventing the loss of data integrity. You can use this descriptive information to do the following:

* Verify data accuracy.
* Automatically customize data presentation for different users
* Store data consistently across multiple platforms.

Improve search efficiency.

Computer programs like search engines can sort and categorize XML files more efficiently and precisely than other types of documents. For example, the word *mark*can be either a noun or a verb. Based on XML tags, search engines can accurately categorize *mark* for relevant search results. Thus, XML helps computers to interpret natural language more efficiently.

**Design flexible applications**

With XML, you can conveniently upgrade or modify your application design. Many technologies, especially newer ones, come with built-in XML support. They can automatically read and process XML data files so that you can make changes without having to reformat your entire database.

**What are the applications of XML?**

Extensible Markup Language (XML) is the underlying technology in thousands of applications, ranging from common productivity tools like word processing to book publishing software and even complex application configuration systems.

**Data transfer**

You can use XML to transfer data between two systems that store the same data in different formats. For example, your website stores dates in MM/DD/YYYY format, but your accounting system stores dates in DD/MM/YYYY format. You can transfer the data from the website to the accounting system by using XML. Your developers can write code that automatically converts the following:

* Website data to XML format
* XML data to accounting system data
* Accounting system data back to XML format
* XML data back to website data

**Web applications**

XML gives structure to the data that you see on webpages. Other website technologies, like HTML, work with XML to present consistent and relevant data to website visitors. For example, consider an e-commerce website that sells clothes. Instead of showing all clothes to all visitors, the website uses XML to create customized webpages based on user preferences. It shows products from specific brands by filtering the <brand> tag.

**Documentation**

You can use XML to specify the structural information of any technical document. Other programs then process the document structure to present it flexibly. For example, there are XML tags for a paragraph, an item in a numbered list, and a heading. Using these tags, other types of software automatically prepare the document for uses such as printing and webpage publication.

**Data type**

Many programming languages support XML as a data type. With this support, you can easily write programs in other languages that work directly with XML files.

**What are the components of an XML file?**

An Extensible Markup Language (XML) file is a text-based document that you can save with the .xml extension. You can write XML similar to other text files. To create or edit an XML file, you can use any of the following:

* Text editors like Notepad or Notepad++
* Online XML editors
* Web browsers

Any XML file includes the following components.

**XML document**

The <xml></xml> tags are used to mark the beginning and end of an XML file. The content within these tags is also called an XML document. It is the first tag that any software will look for to process XML code.

**XML declaration**

An XML document begins with some information about XML itself. For example, it might mention the XML version that it follows. This opening is called an XML declaration. Here's an example.

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

**XML elements**

All the other tags you create within an XML document are called XML elements. XML elements can contain these features:

* Text
* Attributes
* Other elements

All XML documents begin with a primary tag, which is called the root element.

For example, consider the XML file below.

<InvitationList>

<family>

       <aunt>

       <name>Christine</name>

        <name>Stephanie</name>

       </aunt>

</family>

</InvitationList>

<InvitationList> is the root element; *family*and *aunt*are other element names.

**XML attributes**

XML elements can have other descriptors called attributes. You can define your own attribute names and write the attribute values within quotation marks as shown below.

<person age=“22”>

**XML content**

The data in XML files is also called XML content. For example, in the XML file, you might see data like this.

<friend>

       <name>Charlie</name>

       <name>Steve</name>

</friend>

The data values *Charlie*and *Steve*are the content.

**What is an XML schema?**

An Extensible Markup Language (XML) schema is a document that describes some rules or limits on the structure of an XML file. You can describe these constraints in several different ways, like these:

* Grammatical rules to determine the order of elements.
* Yes or No conditions that the content must satisfy.
* Data types for the content in XML files
* Constraints for data integrity

For example, an XML schema for bookstores might impose constraints like these:

1. A book element will have the attributes *title*and *author*.
2. The book element will be nested under a category element with an attribute name.
3. The price of a book will be a separate element nested under *book*.

To meet these constraints, we will write the XML file as shown below.

<category name=“Technology”>

<book title=“Learning Amazon Web Services”, author=“Mark Wilkins”>

<price>$20</price>

</book>

</category>

XML schemas enforce consistency in how different software applications create and use XML files. Some industries implement XML schemas that are specific to their operations to reduce complexity in writing XML code for interbusiness data transfer. For example, Scalable Vector Graphics (SVG) is an XML specification for describing computer graphics-related data. Software developers write XML files so that they meet such industry specifications.

**What is an XML parser?**

An Extensible Markup Language (XML) parser is software that can process or read XML documents to extract the data within them. XML parsers also check the syntax or rules of the XML file and can validate it against a particular XML schema. Because XML is a strict markup language, the parsers will not process the file if there are any validation or syntax errors. For example, the XML parser will give errors if any of these conditions are true:

* A closing tag or end tag is missing.
* Attribute values don’t have quotation marks.
* A schema condition has not been met.

Software applications use XML parsers to transform XML files into native data types. They can thus focus on the application logic without having to go into the details of the XML itself.

**How is XML different from HTML?**

HyperText Markup Language (HTML) is the language used in most webpages. A web browser processes the HTML documents and displays them as a multimedia page. The World Wide Web Consortium (W3C) is the international community that develops protocols and guidelines to ensure the long-term growth of the web. W3C established both the HTML and Extensible Markup Language (XML) standards that website developers implement for consistency and quality.

**XML vs. HTML**

While HTML and XML files look very similar, there are some key differences.

**Purpose**

The purpose of HTML is to present and display data. However, XML stores and transports data.

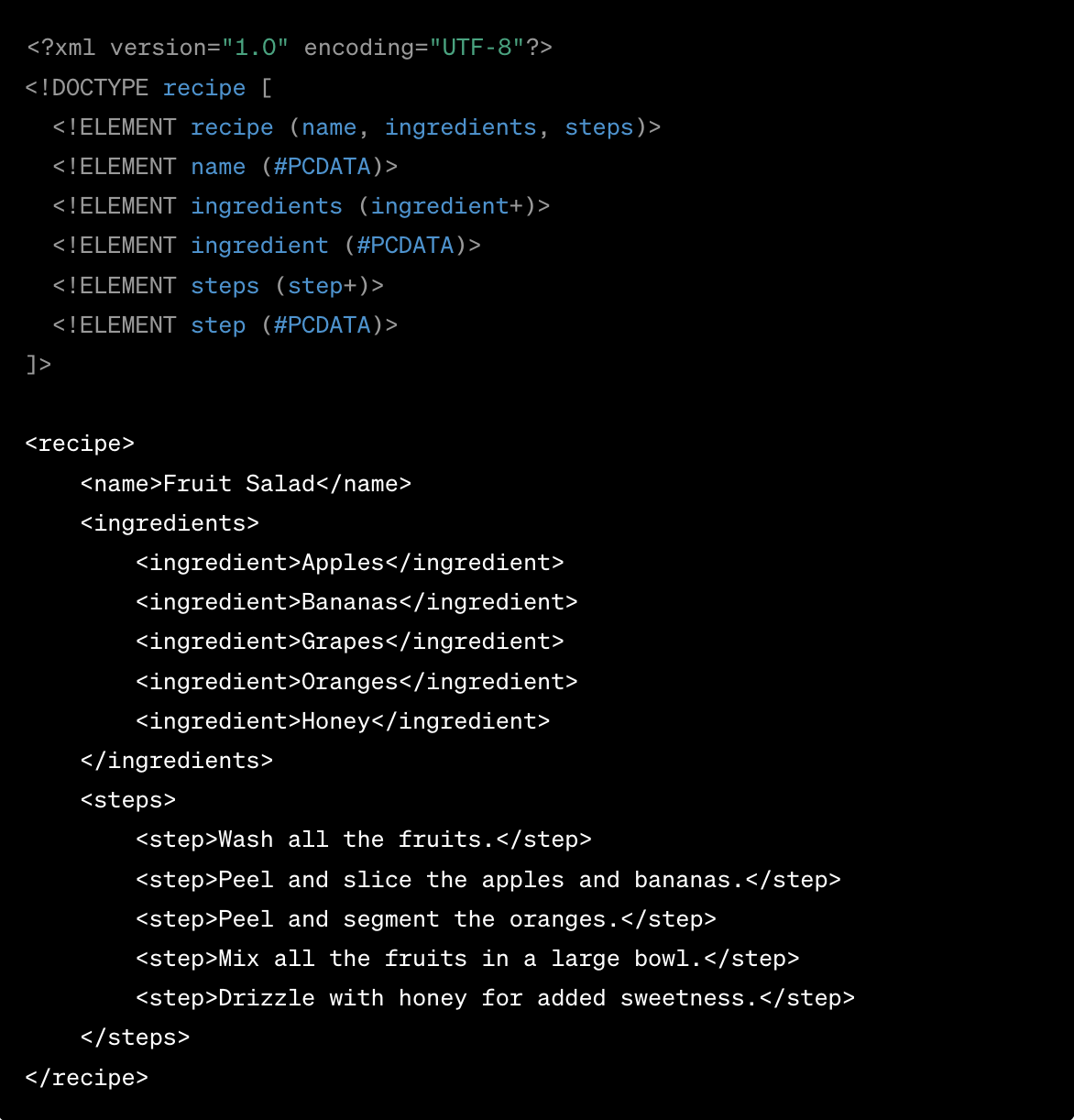
**Tags**

HTML has predefined tags, but users can create and define their own tags in XML.

**Syntax rules**

There are some minor yet important differences between HTML and XML syntax. For example, XML is case sensitive, but HTML is not. XML parsers will give errors if you write a tag as <Book> instead of <book>.

Examples



Here’s a breakdown of what this code means:

**<?xml version="1.0" encoding="UTF-8"?>:** This is the XML declaration. It defines the XML version and the character encoding used in the document.

**<!DOCTYPE ... >:** This is the Document Type Definition (DTD). It defines the structure and the legal elements and attributes of the XML document. In this case, it specifies the structure and elements that can be used in our recipe.

* **<recipe>** contains the entire fruit salad recipe.
* **<name>** tells you the name of the dish.
* **<ingredients>** lists all the items you need.
* **<steps>** provides a sequential guide on how to prepare the fruit salad.