**XML** **(eXtensible Markup Language)** is a text-based format for storing and exchanging structured data on the Internet. In this format, the data is presented as documents with a clear and flexible structure. An XML document can be stored on the computer with the .xml extension that is often used to keep configuration files of programs.

At this moment, XML is one of the most popular formats around the world, used both by small startups and huge companies. XML is especially valued for being very expressive for people and easy for machine processing.

**Tags and elements**

Each XML document consists of **tags**and **elements.**

A **tag** is a string with an assigned meaning like a book, a person or something of the sort. It is interesting that XML does not provide tags at all, but it gives developers an opportunity to invent tags independently.

An **element** is a building block of an XML structure: it may contain text, tags, other elements and attributes.

Here is an example of an XML document that describes a book with a title and one author:

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

<book>

<title>The Three-Body Problem</title>

<author>Liu Cixin</author>

</book>

This document has three tags enclosed in angle brackets: <book>, <title> and <author>.

By element we understand the combination of a starting tag with the corresponding ending tag together with their content. Elements set the structure of a document since they can be nested in each other.

Our document has the following elements:

* <book>....</book> contains two other elements;
* <title>The Three-Body Problem</title>
* <author>Liu Cixin</author>

All elements should have a closing tag (a similar tag, but with a slash / in front) or just end with a slash (/>). Here's an example of using an unpaired tag: <picture name="sun"/>.

The first line in the XML document is called a **prolog:**

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

It specifies the version of the XML standard (usually 1.0) and defines the encoding (here it's UTF-8). The prolog is optional, but if it's there, it must come first in the document.

Note that a prolog does not have a closing tag!

**Child elements**

Each XML document always has a single element called root. This element can contain other elements called **child elements** which in their turn can have their own **children.**

The following XML document represents books contained in a library:

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

<library>

<book>

<title>The Three-Body Problem</title>

<author>Liu Cixin</author>

</book>

<book>

<title>Modern Operating Systems</title>

<author>Andrew S. Tanenbaum</author>

</book>

</library>

Here, the root element <library> has two children <book> elements, while the elements <title> and <author> are the children of <book>. So, XML documents represent hierarchical structures that are often used in programming.

**Attributes**

XML elements can possess **attributes** that provide additional information about the element.

The value of the attribute is always set in either double or single quotes. For example, the name of a picture can be written as follows:

<picture name="The Black Square"/>

If the value of the attribute contains double quotes, you need to use single quotes. For example:

<picture name='"Sunset at Sea", Ivan Aivazovsky'/>

Sometimes, you can also see quotes replaced by special entity symbols (&quot;):

<picture name="&quot;Sunflowers&quot;, Vincent van Gogh"/>

An element can contain more than one attribute:

<picture name='Sunset at Sea' author='Ivan Aivazovsky'/>

The following XML document presents an art gallery:

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

<gallery>

<picture name='Sunset at Sea' painter='Ivan Aivazovsky'/>

<picture name='The Black Square' painter='Kazimir Malevich'/>

<picture name='Sunflowers' painter='Vincent van Gogh'/>

</gallery>

As you can see, in some cases, attributes can replace child elements. There is no consensus about what's better to use. It usually depends on the data you are trying to model, your tools for XML processing and, of course, the people you work with.

Note thatan element can have both attributes and child elements together.

**Pros and cons of XML**

XML has won popularity due to its apparent advantages:

* it can be easily understood by machines and people alike;
* the format is based on international standards;
* it has a well-defined structure which facilitates the search and extraction of information;
* modern programming languages have libraries for processing XML documents automatically.

At the same time, XML has an important disadvantage. Its redundant syntax causes higher storage and transportation cost. This is especially important when we need to store or transfer a large amount of data.

**Conclusion**

In summary, XML is a format for saving and transferring data as documents with the .xml extension. The main components of these documents are tags, elements, and attributes. Keep in mind, though XML has its advantages like a well-defined structure, using it with a large amount of data could sometimes be inefficient due to its verbose syntax.  
Now that you have learned the basic principles of XML, it is time to put your skills to practice!

## XML syntax

Is there something wrong with this syntax?

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

<countries>

<country name="USA" />

<country name="CANADA" />

</countries>

Nothing wrong it is correct

## XML code mistake

Find a mistake in this code snippet:

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

<world>

<country name="USA" />

<country name="CANADA" />

</world>

There are no mistakes

**XML abbreviation**

What does XML stand for?

Extensible Markup Language

**Coordinates as XML**

Sort the code fragments so that they make a valid XML:

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

<location>

<coordinates latitude="90.0000" longitude="0.0000"/>

</location>

**A prologue**

What does this prologue determine?

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

**Correct answers:**

* ✔️ **a version of the XML standard**
* ✔️ **a character encoding**

### Explanation:

The XML prologue:

xml

CopyEdit

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

* version="1.0" → Specifies the **version of the XML standard** being used.
* encoding="UTF-8" → Specifies the **character encoding** used in the document.

**Find the root**

 Report a typo

Here is an XML that describes a library with books:

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

<library>

<books>

<book title="The Three-Body Problem"/>

<book title="Modern Operating Systems"/>

</books>

</library>

What is the root element here? <library>

**Online shop**

Write an XML that represents a set of products in an online store.

The required name for the XML root element is products. It should include two or more nested elements named product.

Each product includes two child elements: name and cost.

You can skip the prologue here.

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

<products>

<product>

<name> product1</name>

<cost>1</cost>

</product>

<product>

<name>product2' </name>

<cost>2</cost>

</product>

</products>

**Person as XML**

Sort the code fragments so that they make a valid XML document

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

<person>

<name>Albert Einstein</name>

</person>

**Your country**

Write an XML that represents your country as a single element named country.

Include name and population as attributes to this element.

You can skip the prologue here.

Be careful. Do not confuse tags with attributes.

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

<country name="Canada" population="38000000"/>

An attribute

Select an element with a correct attribute.

<person age="30"/>

**A book with several authors**

Write an XML document that represents a book with the title attribute and one nested element named authors.

The authors element should have two or more child elements named author with only the name attribute.

You can skip the prologue here.

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

<book title="book1" >

<authors>

<author name="author1"/>

<author name="auhtor2"/>

</authors>

</book>