Maybe it is a little hard to understand, but **XML** does not DO anything. **XML** was created to structure, store, and transport information. Основное назначение это перенос информации.

It is just pure information wrapped in tags. Someone must write a piece of software to send, receive or display it. The tags (like <to> and <from>) are not defined in any XML standard. These tags are "invented" by the author of the XML document.

That is because the XML language has no predefined tags.

The tags used in HTML (and the structure of HTML) are predefined. HTML documents can only use tags defined in the HTML standard (like <p>, <h1>, etc.).

XML allows the author to define his own tags and his own document structure.

XML is Not a Replacement for HTML

XML is a complement (дополнение) to HTML.

It is important to understand that XML is not a replacement for HTML. In most web applications, **XML is used to transport data**, while **HTML is used to format and display the data**.

My best description of XML is this:

**XML is a software and hardware independent tool for carrying information.**

XML documents use a self-describing and simple syntax:

<?xml version="1.0" encoding="ISO-8859-1"?>

<note>

<to>Tove</to>

<from>Jani</from>

<heading>Reminder</heading>

<body>Don't forget me this weekend!</body>

</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"):

<note>

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

<to>Tove</to>

<from>Jani</from>

<heading>Reminder</heading>

<body>Don't forget me this weekend!</body>

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

</note>

XML Documents Form a Tree Structure

XML documents must contain a **root** element. This element is "the parent" of all other elements.

The elements (tegs) 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):

<root>

<child>

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

</child>

</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).

**XML Syntax Rules**

1)-**All XML Elements Must Have a Closing Tag**

In HTML, you will often see elements that don't have a closing tag:

<p>This is a paragraph

<p>This is another paragraph

In XML, it is illegal to omit the closing tag. All elements must have a closing tag:

<p>This is a paragraph</p>

<p>This is another paragraph</p>

Note: You might have noticed from the previous example that the XML declaration did not have a closing tag. This is not an error. The declaration is not a part of the XML document itself, and it has no closing tag.

2)- **XML Tags are Case Sensitive**

XML elements are defined using XML tags.

XML tags are case sensitive. With XML, the tag <Letter> is different from the tag <letter>.

Opening and closing tags must be written with the same case:

<Message>This is incorrect</message>

<message>This is correct</message>

Note: "Opening and closing tags" are often referred to as "Start and end tags". Use whatever you prefer. It is exactly the same thing.

3)-**XML Elements Must be Properly Nested**

<b><i>This text is bold and italic</i></b>

In the example above, "Properly nested" simply means that since the <i> element is opened inside the <b> element, it must be closed inside the <b> element.

4)-**XML Documents Must Have a Root Element**

XML documents must contain one element that is the parent of all other elements. This element is called the root element.

<root>

<child>

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

</child>

</root>

5)-**XML Attribute Values Must be Quoted**

XML elements can have attributes in name/value pairs just like in HTML.

In XML the attribute value must always be quoted. Study the two XML documents below. The first one is incorrect, the second is correct:

<note date=12/11/2007>

<to>Tove</to>

<from>Jani</from>

</note>

<note date="12/11/2007">

<to>Tove</to>

<from>Jani</from>

</note>

The error in the first document is that the **date attribute in the note element is not quoted**.

6)-**Entity References**

Some characters have a special meaning in XML.

If you place a character like "<" inside an XML element, it will generate an error because the parser interprets it as the start of a new element.

This will generate an XML error:

<message>if salary < 1000 then</message>

To avoid this error, replace the "<" character with an entity reference:

<message>if salary &lt; 1000 then</message>

There are 5 predefined entity references in XML:

&lt; < less than

&gt; > greater than

&amp; & ampersand

&apos; ' apostrophe

&quot; " quotation mark

Note: Only the characters "<" and "&" are strictly illegal in XML. The greater than character is legal, but it is a good habit to replace it.

6)-**Comments in XML**

The syntax for writing comments in XML is similar to that of HTML.

<!-- This is a comment -->

With XML, White Space is Preserved

HTML reduces multiple white space characters to a single white space:

HTML: Hello my name is Tove

Output: Hello my name is Tove.

With XML, the white space in your document is not truncated.(не усекается.)

7)-**XML Stores New Line as LF**

In Windows applications, a new line is normally stored as a pair of characters: carriage return (CR) and line feed (LF). The character pair bears some resemblance to the typewriter actions of setting a new line. In Unix applications, a new line is normally stored as a LF character. Macintosh applications use only a CR character to store a new line.

**What is an XML Element? (tag)**

An element can contain other elements, simple text or a mixture of both. Elements can also have attributes.

<bookstore>

<book category="CHILDREN">

<title>Harry Potter</title>

<author>J K. Rowling</author>

<year>2005</year>

<price>29.99</price>

</book>

<book category="WEB">

<title>Learning XML</title>

<author>Erik T. Ray</author>

<year>2003</year>

<price>39.95</price>

</book>

</bookstore>

In the example above, <bookstore> and <book> **have element contents**, because they contain other elements. <author> **has text content** because it contains text.

In the example above only <book> has an **attribute** (category="CHILDREN").

**XML Naming Rules**

XML elements must follow these naming rules:

Names can contain letters, numbers, and other characters

Names cannot start with a number or punctuation character

Names cannot start with the letters xml (or XML, or Xml, etc)

Names cannot contain spaces

Any name can be used, no words are reserved.

Best Naming Practices

Make names descriptive. Names with an underscore separator are nice: <first\_name>, <last\_name>.

Names should be short and simple, like this: <book\_title> not like this: <the\_title\_of\_the\_book>.

Avoid "-" characters. If you name something "first-name," some software may think you want to subtract name from first.

Avoid "." characters. If you name something "first.name," some software may think that "name" is a property of the object "first."

Avoid ":" characters. Colons are reserved to be used for something called namespaces (more later).

XML documents often have a corresponding database. A good practice is to use the naming rules of your database for the elements in the XML documents.

Non-English letters like éòá are perfectly legal in XML, but watch out for problems if your software vendor doesn't support them.

**XML Elements are Extensible(РАСШИРЯЕМЫ, Т.Е. ЭТО ТО, ЧТО ДЕЛАЕТ SOAP)**, нет это не верно речь идёт о том что XML расширяемый так как можно добавлять какие хочеш новые элементы(новые тэги), так как они не предопределены, как в HTML. Другими словами в HTML есть определённое количество (заранее зарезервированных и запрограммированных на строго определенный функционал) тэгов, а в XML такого нет, все тэги создаются разработчиком в процессе и соответственно их разновидность может быть какой угодно, что позволяет добавлять (увеличивать разновидность) их на любом этапе.

XML elements can be extended to carry more information.

Look at the following XML example:

<note>

<to>Tove</to>

<from>Jani</from>

<body>Don't forget me this weekend!</body>

</note>

Let's imagine that we created an application that extracted the <to>, <from>, and <body> elements from the XML document to produce this output:

MESSAGE

To: Tove

From: Jani

Don't forget me this weekend!

Imagine that the author of the XML document added some extra information to it:

<note>

<date>2008-01-10</date>

<to>Tove</to>

<from>Jani</from>

<heading>Reminder</heading>

<body>Don't forget me this weekend!</body>

</note>

Should the application break or crash?

**No**. The application should still be able to find the <to>, <from>, and <body> elements in the XML document and produce the same output.

**One of the beauties of XML, is that it can often be extended without breaking applications.**

**XML Attributes**

XML elements can have attributes **in the start tag**, (т.е. в открывающем теге) just like HTML.

Attributes provide additional information about elements.

XML Attributes

From HTML you will remember this: <img src="computer.gif">. The "src" attribute provides additional information about the <img> element.

In HTML (and in XML) attributes provide additional information about elements:

<img src="computer.gif">

<a href="demo.asp">

Attributes often provide information that is not a part of the data. In the example below, the file type is irrelevant(не имеющий отношения) to the data, but important to the software that wants to manipulate the element:

<file type="gif">computer.gif</file>

XML Attributes values Must be Quoted

Attribute values must always be enclosed in quotes, but either single or double quotes can be used. For a person's sex, the person tag can be written like this: <person sex="female">

or like this: <person sex='female'>

If the attribute value itself contains double quotes you can use single quotes, like in this example:

<gangster name=**'George "Shotgun" Ziegler'**>

or you can use character entities: <gangster name="George &quot;Shotgun&quot; Ziegler">

XML Elements vs. Attributes

Take a look at these examples:

<person sex="female">

<firstname>Anna</firstname>

<lastname>Smith</lastname>

</person>

<person>

<sex>female</sex>

<firstname>Anna</firstname>

<lastname>Smith</lastname>

</person>

In the first example sex is an attribute. In the last, sex is an element. Both examples provide the same information.

There are no rules about when to use attributes and when to use elements. Attributes are handy in HTML. In XML my advice is to avoid them. Use elements instead.

My Favorite Way

The following three XML documents contain exactly the same information:

A date attribute is used in the first example:

<note date="10/01/2008">

<to>Tove</to>

<from>Jani</from>

<heading>Reminder</heading>

<body>Don't forget me this weekend!</body>

</note>

A date element is used in the second example:

<note>

<date>10/01/2008</date>

<to>Tove</to>

<from>Jani</from>

<heading>Reminder</heading>

<body>Don't forget me this weekend!</body>

</note>

An expanded date element is used in the third: (THIS IS MY FAVORITE):

<note>

<date>

<day>10</day>

<month>01</month>

<year>2008</year>

</date>

<to>Tove</to>

<from>Jani</from>

<heading>Reminder</heading>

<body>Don't forget me this weekend!</body>

</note>

Avoid XML Attributes?

Some of the problems with using attributes are:

attributes cannot contain multiple values (elements can)

attributes cannot contain tree structures (elements can)

attributes are not easily expandable (for future changes)

Attributes are difficult to read and maintain. Use elements for data. Use attributes for information that is not relevant to the data.

Don't end up like this:

<note day="10" month="01" year="2008"

to="Tove" from="Jani" heading="Reminder"

body="Don't forget me this weekend!">

</note>

XML Attributes for Metadata

Sometimes ID references are assigned to elements. These IDs can be used to identify XML elements in much the same way as the ID attribute in HTML. This example demonstrates this:

<messages>

<note id="501">

<to>Tove</to>

<from>Jani</from>

<heading>Reminder</heading>

<body>Don't forget me this weekend!</body>

</note>

<note id="502">

<to>Jani</to>

<from>Tove</from>

<heading>Re: Reminder</heading>

<body>I will not</body>

</note>

</messages>

The ID above is just an identifier, to identify the different notes. It is not a part of the note itself.

What I'm trying to say here is that metadata (data about data) should be stored as attributes, and that data itself should be stored as elements.

**SOAP XML (not WSDL)**

It is important for web applications to be able to communicate over the Internet.

The best way to communicate between applications is over HTTP, because HTTP is supported by all Internet browsers and servers. SOAP was created to accomplish this.

SOAP provides a way to communicate between applications running on different operating systems, with different technologies and programming languages.

A SOAP message is encoded as an XML document, consisting of an <Envelope> element, which contains an optional <Header> element, and a mandatory <Body> element. The <Fault> element, contained in the <Body>, is used for reporting errors.

**The SOAP envelope**

The SOAP <Envelope> is the root element in every SOAP message. It contains two child elements, an optional <Header>, and a mandatory <Body>.

**The SOAP header**

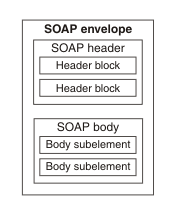
The SOAP <Header> is an optional subelement of the SOAP envelope. It is used to pass application-related information that is to be processed by SOAP nodes along the message path.

**The SOAP body**

The SOAP <Body> is a mandatory subelement of the SOAP envelope. It contains information intended for the ultimate recipient (конечный получатель)of the message.

**The SOAP fault**

The SOAP <Fault> is a subelement of the SOAP body, which is used for reporting errors.



With the exception of the <Fault> element, which is contained in the <Body> of a SOAP message, XML elements in the <Header> and the <Body> are defined by the applications that make use of them.

This diagram shows the structure of a SOAP message, which is described in the text.

**WSDL Documents**

The three major elements of WSDL that can be defined separately are −

-Types

-Operations

-Binding

A WSDL document has various elements, but they are contained within these three main elements

An WSDL document describes a web service. It specifies the location of the service, and the methods of the service, using these major elements(tags):

Element: Description:

<types> Defines the (XML Schema) data types used by the web service. a container for ` data type definitions using some type system (such as XSD).

A web service needs to define its inputs and outputs and how they are mapped into and out of the services. WSDL <types> element takes care of defining the data types that are used by the web service. Types are XML documents, or document parts.

The types element describes all the data types used between the client and the server.

WSDL is not tied exclusively to a specific typing system.

WSDL uses the W3C **XML Schema specification** as its default choice to define data types.

If the service uses only XML Schema built-in simple types, such as strings and integers, then types element is not required.

WSDL allows the types to be defined in separate elements so that the types are reusable with multiple web services.

<message> typed definition of the data being communicated.

<operation> description of an action supported by the service

<portType> Describes the operations that can be performed and the messages involved.

<binding> Defines the protocol and data format for each port type

<port> a single endpoint defined as a combination of a binding and a network address

<service> a collection of related endpoints.

The main structure of a **WSDL** document looks like this:

<definitions>

<types>

data type definitions........

</types>

<message>

definition of the data being communicated....

</message>

<portType>

set of operations......

</portType>

<binding>

protocol and data format specification....

</binding>

</definitions>

**XSD**

XML Schema Definition Language. XML Schemas define the elements (tags) of your XML files.

What is a Simple Element?

A simple element is an XML element that can contain only text. It cannot contain any other elements or attributes. However, the "only text" restriction is quite misleading. The text can be of many different types. It can be one of the types included in the XML Schema definition (boolean, string, date, etc.), or it can be a custom type that you can define yourself. You can also add restrictions (facets) to a data type in order to limit its content, or you can require the data to match a specific pattern.

The syntax for defining a simple element is:

<xs:element name="xxx" type="yyy"/>

where xxx is the name of the element and yyy is the data type of the element.

XML Schema has a lot of built-in data types. The most common types are:

xs:string

xs:decimal

xs:integer

xs:boolean

xs:date

xs:time

XSD Example:

<?xml version="1.0"?>

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

<xs:element name="note">

<xs:complexType>

<xs:sequence>

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

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

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

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

</xs:sequence>

</xs:complexType>

</xs:element>

</xs:schema>

The purpose of an XML Schema is to define the legal building blocks of an XML document:the 1)-elements and attributes that can appear in a document

2)-the number of (and order of) child elements

3)-data types for elements and attributes

4)-default and fixed values for elements and attributes

XML Schemas are written in XML:

-it must begin with the XML declaration

-it must have one unique root element

-start-tags must have matching end-tags

-elements are case sensitive

-all elements must be closed

-all elements must be properly nested

-all attribute values must be quoted

-entities must be used for special characters

**! ! ! XML documents can have a reference to a DTD or to an XML Schema. ! ! !**

Look at this simple XML document called "**note.xml**":

<?xml version="1.0"?>

<note>

<to>Tove</to>

<from>Jani</from>

<heading>Reminder</heading>

<body>Don't forget me this weekend!</body>

</note>

**A DTD File**

The following example is a DTD file called "**note.dtd**" that **defines** the elements of the XML document above in "note.xml":

<!ELEMENT note (to, from, heading, body)>#1

<!ELEMENT to (#PCDATA)> #2

<!ELEMENT from (#PCDATA)> #3

<!ELEMENT heading (#PCDATA)> #4

<!ELEMENT body (#PCDATA)> #5

**The first line defines the note element to have four child elements: "to, from, heading, body".**

**Line 2-5 defines the to, from, heading, body elements to be of type "#PCDATA".**

**A Reference to a DTD**

This XML document has a reference to a DTD:

<?xml version="1.0"?>

<!DOCTYPE note SYSTEM

"https://www.w3schools.com/xml/note.dtd">

<note>

<to>Tove</to>

<from>Jani</from>

<heading>Reminder</heading>

<body>Don't forget me this weekend!</body>

</note>

**An XML Schema**

The following example is an XML Schema file called "**note.xsd**" that defines the elements of the XML document above in "note.xml":

<?xml version="1.0"?>

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

targetNamespace="https://www.w3schools.com"

xmlns="https://www.w3schools.com"

elementFormDefault="qualified">

<xs:element name="note">

<xs:complexType> <--The note element is a complex type because it contains other elements

<xs:sequence>

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

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

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

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

</xs:sequence>

</xs:complexType>

</xs:element>

</xs:schema>

The other elements (to, from, heading, body) are simple types because they do not contain other elements.

**- The <schema>** element is the root element of every XML Schema, The <schema> element may contain some attributes:

<?xml version="1.0"?>

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

targetNamespace="https://www.w3schools.com"

xmlns="https://www.w3schools.com"

elementFormDefault="qualified">

...

...

</xs:schema>

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

indicates that the elements and data types used in the schema come from the "http://www.w3.org/2001/XMLSchema" namespace. It also specifies that the elements and data types that come from the "http://www.w3.org/2001/XMLSchema" namespace should be prefixed with xs:

targetNamespace="https://www.w3schools.com"

indicates that the elements defined by this schema (note, to, from, heading, body.) come from the "https://www.w3schools.com" namespace.

xmlns="https://www.w3schools.com"

indicates that the default namespace is "https://www.w3schools.com".

elementFormDefault="qualified"

indicates that any elements used by the XML instance document which were declared in this schema must be namespace qualified.

**A Reference to an XML Schema**

This XML document has a reference to an XML Schema:

<?xml version="1.0"?>

<note xmlns="https://www.w3schools.com"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="https://www.w3schools.com note.xsd">

<to>Tove</to>

<from>Jani</from>

<heading>Reminder</heading>

<body>Don't forget me this weekend!</body>

</note>

xmlns="https://www.w3schools.com"

specifies the default namespace declaration. This declaration tells the schema-validator that all the elements used in this XML document are declared in the "https://www.w3schools.com" namespace.

Once you have the XML Schema Instance namespace available:

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

you can use the schemaLocation attribute. This attribute has two values, separated by a space. The first value is the namespace to use. The second value is the location of the XML schema to use for that namespace:

xsi:schemaLocation="https://www.w3schools.com note.xsd"