UNIVERSITÀ
DI PARMA

# XML

***Tecnologie Internet***
a.a. 2022/2023

***Michele Amoretti***

## Introduction

XML stands for **EXtensible Markup Language**.

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

XML is not a replacement for HTML.
XML and HTML were designed with different goals:

- XML was designed to describe data and to store data
- HTML was designed to display data

## Introduction

```
<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

The tags in the example above (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 are predefined. HTML documents can only use tags defined in the HTML standard (like <P>, <H1>, etc.).
XML allows the author to define his/her own tags and his/her own document structure.

## When to use XML

**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 concentrate 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.

## When to use XML

**XML Simplifies Data Storing**

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.

**XML Simplifies Data Sharing**

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.

## When to use XML

**XML Simplifies Platform Changes**

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.

**Internet Languages Written in XML**
Several Internet languages are written in XML. Here are some examples:

- Scalable Vector Graphics (SVG)
- Web Service Description Language (WSDL)
- RDF Site Summary (RSS)

## XML tree

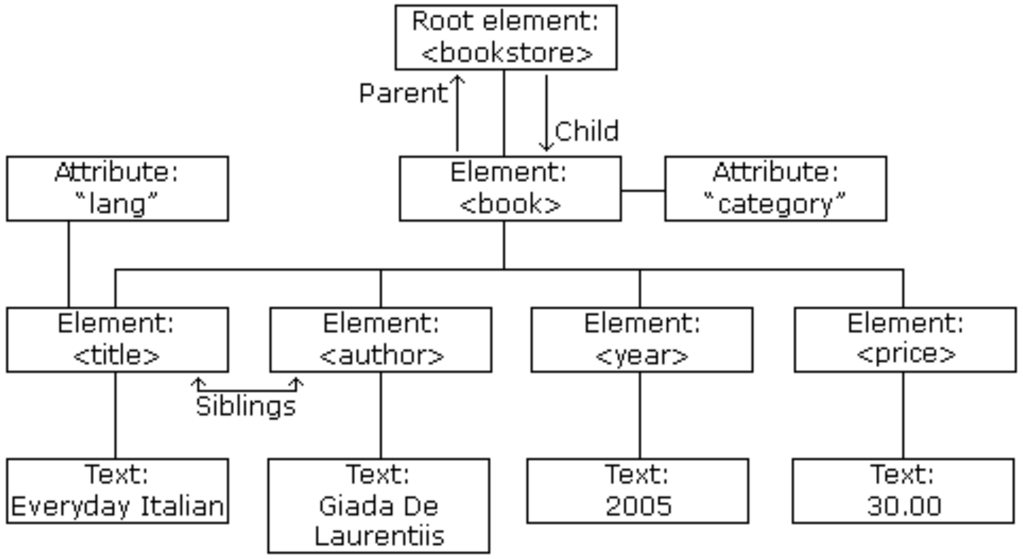
XML documents form a **tree structure** that starts at "the root" and branches to "the leaves".

```
<?xml version="1.0" encoding="UTF-8"?>
<root>
  <child>
    <subchild>.....</subchild>
  </child>
</root>
```

The **prolog** defines the XML version (1.0).
The tree starts at the **root element** and branches to the lowest level of the tree. All elements can have sub elements (**child elements**). Children on the same level are called **siblings** (brothers or sisters).
All elements can have text content and attributes (just like in HTML).

## XML tree

Example:



```xml
<bookstore>
  <book category="COOKING">
    <title lang="en">Everyday Italian</title>
    <author>Giada De Laurentiis</author>
    <year>2005</year>
    <price>30.00</price>
  </book>
  <book category="CHILDREN">
    <title lang="en">Harry Potter</title>
    <author>J K. Rowling</author>
    <year>2005</year>
    <price>29.99</price>
  </book>
  <book category="WEB">
    <title lang="en">Learning XML</title>
    <author>Erik T. Ray</author>
    <year>2003</year>
    <price>39.95</price>
  </book>
</bookstore>
```

## XML syntax

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

```
<p>This is a paragraph.</p>
<br />
```

**Exception: the prolog** is not a part of the XML document itself, and it has no closing tag.

**XML tags are case sensitive.** 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>
```

In XML, all elements must be properly nested within each other:

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

## XML syntax

XML elements can have **attributes** in name/value pairs just like in HTML.
In XML, the attribute values must always be quoted.

```
<note date="12/11/2007">
  <to>Tove</to>
  <from>Jani</from>
</note>
```

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

<!-- This is a comment -->

XML does not truncate multiple white-spaces in a document (while HTML truncates multiple white-spaces to one single white-space).

XML stores a new line as LF.

UNIVERSITÀ DI PARMA

## XML syntax

Some characters have a special meaning in XML. 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>
```

Also:

&gt; for >
&amp; for &
&apos; for �trying
&quot; for “

## XML elements

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

- other elements
- text
- attributes
- or a mix of all of the above...

In the example above, <bookstore> and <book> have **element contents**, because they contain other elements. <book> also has an **attribute** (category="CHILDREN"). <title>, <author>, <year>, and <price> have **text content** because they contain text.

Empty elements:

```
<element></element>
<element />
```

## XML elements

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). However, it is preferable to create short descriptive names, like <person>, <firstname>, <lastname>, <book_title>, etc. and to avoid "-", "." and ":".

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

## XML elements

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!

## XML elements

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 be extended without breaking applications.**

UNIVERSITÀ DI PARMA

## XML attributes

XML elements can have attributes, just like HTML.
Attributes provide additional information about an element.

**Attributes often provide information that is not a part of the data.** In the example below, the file type is irrelevant to the data, but can be important to the software that wants to manipulate the element:

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

The following examples are equivalent:

```
<person gender="female">
  <firstname>Anna</firstname>
  <lastname>Smith</lastname>
</person>
```

```
<person>
  <gender>female</gender>
  <firstname>Anna</firstname>
  <lastname>Smith</lastname>
</person>
```

There are no rules about when to use attributes or when to use elements.

## 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.

## XML namespaces

In XML, element names are defined by the developer. This often results in a conflict when trying to mix XML documents from different XML applications.

```
<table>
  <tr>
    <td>Apples</td>
    <td>Bananas</td>
  </tr>
</table>
```

```
<table>
  <name>Pine Table</name>
  <width>80</width>
  <length>120</length>
</table>
```

If these XML fragments were added together, there would be a name conflict. Both contain a <table> element, but the elements have different content and meaning.

## XML namespaces

Name conflicts in XML can easily be avoided using a name prefix.

```
<h:table>
  <h:tr>
    <h:td>Apples</h:td>
    <h:td>Bananas</h:td>
  </h:tr>
</h:table>

<f:table>
  <f:name>Pine Table</f:name>
  <f:width>80</f:width>
  <f:length>120</f:length>
</f:table>
```

When using prefixes in XML, a so-called **namespace** for the prefix must be defined.

The namespace is defined by the **xmlns attribute** in the start tag of an element.

## XML namespaces

The namespace declaration has the following syntax:

xmlns:*prefix*="*URI*"

```
<root>

<h:table xmlns:h="http://www.w3.org/TR/html4/">
  <h:tr>
    <h:td>Apples</h:td>
    <h:td>Bananas</h:td>
  </h:tr>
</h:table>

<f:table xmlns:f="http://www.w3schools.com/furniture">
  <f:name>Pine Table</f:name>
  <f:width>80</f:width>
  <f:length>120</f:length>
</f:table>

</root>
```

## XML namespaces

In the example above, the xmlns attribute in the <table> tag give the h: and f: prefixes a qualified namespace.

When a namespace is defined for an element, all child elements with the same prefix are associated with the same namespace.

Namespaces can be declared in the elements where they are used or in the XML root element:

```
<root xmlns:h="http://www.w3.org/TR/html4/"
xmlns:f="http://www.w3schools.com/furniture">

<h:table>
  <h:tr>
    <h:td>Apples</h:td>
    <h:td>Bananas</h:td>
  </h:tr>
</h:table>
...
```

## XML namespaces

Defining a default namespace for an element saves us from using prefixes in all the child elements. It has the following syntax:

xmlns="*namespaceURI*"

Thus, the previous exaple may become:

```
<table xmlns="http://www.w3.org/TR/html4/">
  <tr>
    <td>Apples</td>
    <td>Bananas</td>
  </tr>
</table>

<table xmlns="http://www.w3schools.com/furniture">
  <name>Pine Table</name>
  <width>80</width>
  <length>120</length>
</table>
```

## XML encoding

XML documents can contain international characters, like Norwegian æøå, or French êèé.

Use the prolog to specify the encoding:

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

UTF = Unicode Transformation Format

The XML standard states that all XML software must understand both UTF-8 and UTF-16.

UTF-8 is the default for documents without encoding information, because it is the Web standard.

In addition, most XML software systems understand encodings like ISO-8859-1, Windows-1252, and ASCII.

## XML display

Raw XML files can be viewed in all major browsers.

Don't expect XML files to be displayed as HTML pages.

```
<?xml version="1.0" encoding="UTF-8"?>
 - <note>
      <to>Tove</to>
      <from>Jani</from>
      <heading>Reminder</heading>
      <body>Don't forget me this weekend!</body>
   </note>
```

Notice that an XML document will be displayed with color-coded root and child elements. A plus (+) or minus sign (-) to the left of the elements can be clicked to expand or collapse the element structure.

## XML display

We can use an XML file like

http://www.w3schools.com/xml/cd_catalog.xml

and a style sheet like

http://www.w3schools.com/xml/cd_catalog.css

RESULT:

http://www.w3schools.com/xml/cd_catalog_with_css.xml

## XML document types

An XML document with correct syntax is called "Well Formed".

A "Valid" XML document must also conform to a **document type definition**.

There are two different document type definitions that can be used with XML:

- **DTD** - The original Document Type Definition
- **XML Schema** - An XML-based alternative to DTD

## XML document types

**When to use a DTD/Schema?**

1. Independent groups of people can agree to use a standard DTD/Schema for interchanging data.

2. Your application can use a standard DTD/Schema to verify that the data you receive from the outside world is valid.

3. You can also use a DTD/Schema to verify your own data.

UNIVERSITÀ
DI PARMA

## XML document types

**When to NOT to Use a DTD/Schema?**

In principle, XML does not require a DTD/Schema.

When you are experimenting with XML, or when you are working with small XML files, creating DTDs may be a waste of time.

If you develop applications, wait until the specification is stable before you add a document definition. Otherwise, your software might stop working because of validation errors.

## XML DTD

The purpose of a DTD is to define the structure of an XML document. It defines the structure with a list of legal elements:

```
<!DOCTYPE note
[
<!ELEMENT note (to,from,heading,body)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT heading (#PCDATA)>
<!ELEMENT body (#PCDATA)>
]>
```

#PCDATA means parse-able text data

The DTD above is interpreted like this:

- !DOCTYPE note defines that the root element of the document is note
- !ELEMENT note defines that the note element must contain four elements: "to, from, heading, body"
- !ELEMENT to defines the to element to be of type "#PCDATA"
- !ELEMENT from defines the from element to be of type "#PCDATA"
- !ELEMENT heading defines the heading element to be of type "#PCDATA"
- !ELEMENT body defines the body element to be of type "#PCDATA"

## XML DTD

A doctype declaration can also be used to define special characters and character strings, used in the document:

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

<!DOCTYPE note [
<!ENTITY nbsp "&#xA0;">
<!ENTITY writer "Writer: Donald Duck.">
<!ENTITY copyright "Copyright: W3Schools.">
]>

<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
<footer>&writer;&nbsp;&copyright;</footer>
</note>
```

## XML Schema

XML Schema is an XML-based alternative to DTD.

```
<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>
```

**XML Schemas are much more powerful than DTD:**

- XML Schemas are written in XML
- XML Schemas are extensible to additions
- XML Schemas support data types
- XML Schemas support namespaces

## References

http://www.w3.org/XML/

https://www.w3schools.com/XML/

https://www.w3.org/TR/REC-xml-names/