

WEB TECHNOLOGY

1ST SEMESTER 2019

Extensible Markup Language

Asst. Prof. Manop Phankokkruad, Ph.D.

Faculty of Information Technology,
King Mongkut's Institute of Technology Ladkrabang, Thailand

Outline

- ❑ What is XML?
- ❑ XML Structure & Syntax
- ❑ Document Type Definition
- ❑ XML Schema
- ❑ XML DOM

Introduction

XML and JSON are the two most common formats for data interchange in the Web today.

Although their purposes are not identical, they are frequently used to accomplish the same task, which is data interchange.

Both have well-documented open standards on the Web (RFC 7159, RFC 4825), and both are human and machine-readable. Neither one is absolutely superior to the other, as each is better suited for different use cases.

What is XML ?

“ eXtensible Markup Language(XML) is set of rules for defining and representing data as structured documents for applications on the Internet. ”

- ❑ XML is a restricted form of SGML (Standard Generalized Markup Language).
- ❑ XML is designed to describe data.
- ❑ XML is compatible with major Internet transmission protocols and is also highly compressible for faster transmission.
- ❑ Almost all major software vendors fully support the general XML standard.

What is XML ?

- ❑ The XML standard is designed to be independent of vendor, operating system, source application, destination application, database, and transport protocol.
- ❑ XML Tags are added to the document to provide the extra information.
- ❑ XML tags are not predefined unlike HTML.
- ❑ XML tags give a reader some idea what some of the data means.
- ❑ There are 3 main components for XML content: XML, DTD and XML Schema define rules to describe data.

Advantages of Using XML

- ❑ XML is text (Unicode) based.
 - Takes up less space.
 - Can be transmitted efficiently.
 - Easily readable by human users
- ❑ One XML document can be displayed differently in different media.
- ❑ XML documents can be modularized.
Parts can be reused.
- ❑ Very flexible and customizable (no finite tag set)
- ❑ Easy to convert into other representations (XML transformation languages)

XML Structure

Document Prolog comes at the top of the document, before the root element. This section contains XML declaration and Document type declaration. Document Elements are the building blocks of XML.

```
<?xml version="1.0"?>
```

```
<root>
```

```
  <element1></element1>
```

```
  <element2 type="string"></element2>
```

```
  <element3 type="integer" value="9.3"></element3>
```


```
</root>
```


Document Elements

Document Prolog

Building Blocks of XML

Elements (Tags) are the primary components of XML documents.

Element author with Attribute id  `<?xml version="1.0"/>`
`<author id = "0123">`

Element fname nested inside element author.  `<fname>Manop</fname>`
`<lname>Phankokkruad</lname>`
`<phone>212-346-1234</phone>`
`</author>`

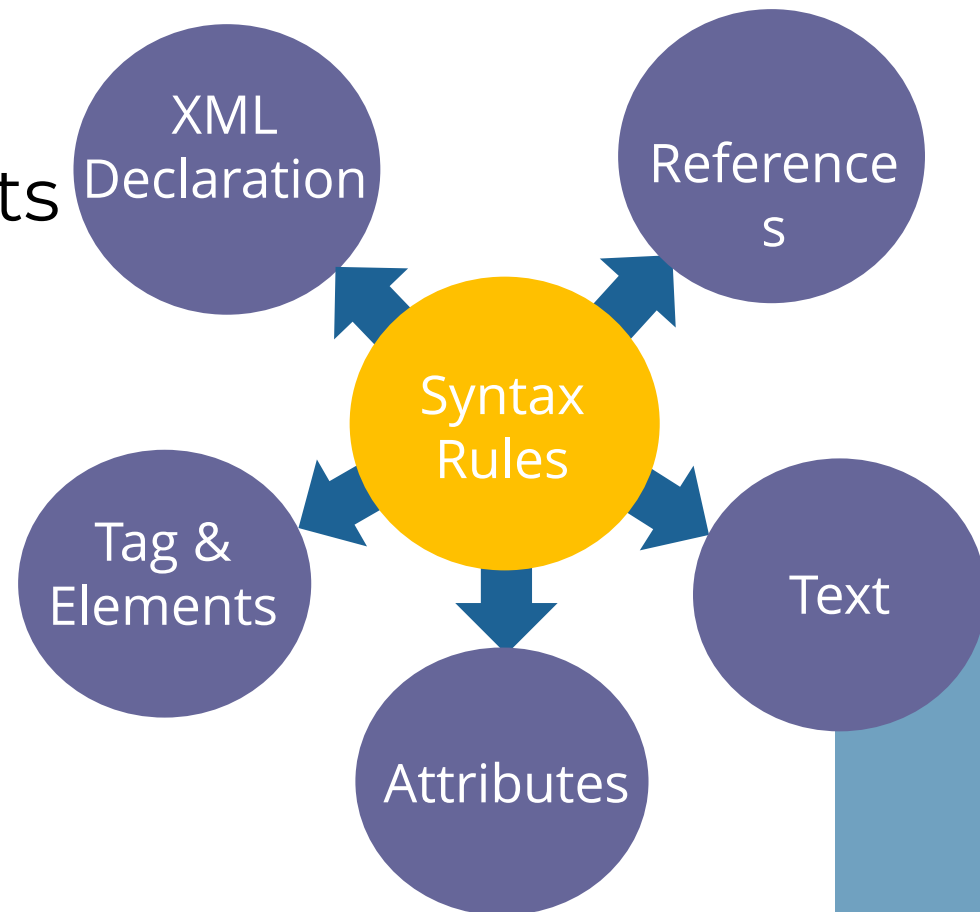
`<!-- I am comment -->`

- ❑ Attributes provide additional information about Elements. Values of the Attributes are set inside the Elements .
- ❑ Comments starts with `<!--` and end with `-->`.

XML – Syntax

A diagram depicts the syntax rules to write different types of markup and text in an XML document.

1. XML Declaration
2. Tags and Elements
3. XML Attributes
4. XML References
5. XML Text



XML Declaration

The XML document can optionally have an XML declaration. It is written as follows:

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

Syntax Rules for XML Declaration

- The XML declaration is case sensitive and must begin with **<?xml>** where **xml** is written in lower-case.
- It strictly needs to be the first statement of the XML document.
- The XML declaration strictly needs be the first statement in the XML document.

Tags and Elements

An XML file is structured by several XML-elements, also called XML-nodes or XML-tags. The names of XML-elements are enclosed in triangular brackets “< >”.

Syntax Rules for Tags and Elements

- **Element Syntax:** Each XML-element needs to be closed either with start or with end elements.

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

- **Nesting of Elements:** An XML-element can contain multiple XML-elements as its children, but the children elements must not overlap.

Tags and Elements (cont.)

Root Element: An XML document can have only one root element.

```
<root>  
  <x>...</x>  
  <y>...</y>  
</root>
```

Case Sensitivity: The names of XML-elements are case-sensitive. That means the name of the start and the end elements need to be exactly in the same case.

For example, **<contact-info>** is different from **<Contact-Info>**.

XML Attributes

An attribute specifies a single property for the element, using a name/value pair. An XML element can have one or more attributes.

```
<person gender="female"> ... </person>
```

Syntax Rules for XML Attributes

- Attribute names in XML are case sensitive.
- Same attribute cannot have two values in a syntax.
- Attribute names are defined without quotation marks, whereas attribute values must always appear in quotation marks.

XML References

References usually allow you to include additional text or markup in an XML document. References always begin with the symbol "&" which is a reserved character and end with the symbol ";".

- **Entity References:** An entity reference contains a name between the start and the end delimiters.
- **Character References:** These contain references, such as A. In this case, 65 refers to alphabet "A".

XML References

An example of XML References

Character	Predeclared Entity
&	&
<	<
>	>
"	"
'	'

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

```
<gangster name="George  
&quot;Shotgun&quot; Ziegler">
```



XML Text

The names of XML-elements and XML-attributes are case-sensitive. To avoid character encoding problems, all XML files should be saved as **Unicode UTF-8 or UTF-16** files.

Whitespace characters like blanks, tabs and line-breaks between XML-elements and between the XML-attributes will be ignored.

Some characters are reserved by the XML syntax itself. To use them, some replacement-entities are used.

Validity

- ❑ A well-formed document has a tree structure and obeys all the XML rules.
- ❑ A particular application may add more rules in either a DTD (document type definition) or in a schema.
- ❑ Many specialized DTDs and schemas have been created to describe particular areas.

Validity

- ❑ An XML document with correct syntax is called **well-formed** document.
 - XML documents must have a root element
 - XML elements must have a closing tag
 - XML tags are case sensitive
 - XML elements must be properly nested
 - XML attribute values must be quoted

Document Type Definitions

3

A Document Type Definitions (DTD) describes the tree structure of a document and something about its data.

A DTD defines the structure and the legal elements and attributes of an XML document. An application can use a DTD to verify that XML data is valid.

There are two data types, PCDATA and CDATA.

- **PCDATA** is parsed character data.
- **CDATA** is character data, not usually parsed.

DTD Example

XML Document:

```
<?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>
```

□ DTD:

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

DTD Declaration

XML document with an internal DTD

```
<?xml version="1.0"?>
```

```
<!DOCTYPE note [  
  <!ELEMENT note (to,from,heading,body)>  
  <!ELEMENT to (#PCDATA)>  
  <!ELEMENT from (#PCDATA)>  
  <!ELEMENT heading (#PCDATA)>  
  <!ELEMENT body (#PCDATA)>  
<note>  
  <to>Tove</to>  
  <from>Jani</from>  
  <heading>Reminder</heading>  
  <body>Don't forget me this weekend</body>  
</note>
```

DTD Declaration

XML document with a reference to an external DTD

```
<?xml version="1.0"?>  
<!DOCTYPE note SYSTEM "note.dtd">  
<note>  
  <to>Tove</to>  
  <from>Jani</from>  
  <heading>Reminder</heading>  
  <body>Don't forget me this weekend!</body>  
</note>
```

XML Schema



An XML Schema describes the structure of an XML document. The XML Schema language is also referred to as **XML Schema Definition (XSD)**.

- ❑ XSD is used to describe and validate the structure and the content of XML data.
- ❑ XML Schema defines the **elements, attributes and data types**.
- ❑ XML Schemas are written in XML document.
- ❑ They were standardized after DTDs and provide more information about the document.

XML Schema

- ❑ Serves same purpose as database schema.
- ❑ **Data types** including string, decimal, integer, boolean, date, and time.
- ❑ They divide **elements** into simple and complex types.
- ❑ Allows creation of user-defined complex types.
- ❑ They also determine the tree structure and how many children a node may have.

XML Schema

□ Definition Types

- *Simple type element* is used only in the context of the text. Some of the predefined simple types are: xs:integer, xs:boolean, xs:string, xs:date. For example

```
<xs:element name = "phone_number" type = "xs:int" />
```

XML Schema

□ Definition Types (cont.)

- A *complex type* is a container for other element definitions. This allows you to specify which child elements an element can contain and to provide some structure within your XML documents. For example

```
<xs:element name = "Address">
  <xs:complexType>
    <xs:sequence>
      <xs:element name = "name" type = "xs:string" />
      <xs:element name = "company" type = "xs:string" />
      <xs:element name = "phone" type = "xs:int" />
    </xs:sequence>
  </xs:complexType>
```

XML Schema

□ Attributes

- *Attributes* in XSD provide extra information within an element. Attributes have name and type property as shown below;

```
<xs:attribute name = "x" type = "y"/>
```

XML Schema Example

□ XML Schema

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="../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>
```

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

XML Document

More XML Schema Example

□ XML Schema

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="../XMLSchema">
  <xs:complexType name = "StudentType">
    <xs:attribute name="id" type="xs:string"/>
    <xs:element name="Name" type="xs:string />
    <xs:element name="Age" type="xs:integer"/>
    <xs:element name="Email" type="xs:string"/>
  </xs:complexType>
</xs:schema>
```

```
<Students>
  <Student id="p1">
    <Name>Allan</Name>
    <Age>26</Age>
    <Email>allan@abc.com</Email>
  </Student>
</Students>
```

XML Document

XML Advantages

1. Meta data support : One of the biggest advantage of XML is that can put metadata into the tags in the form of attributes.
2. Browser rendering : The most browsers render it in a highly readable and organized way. The tree structure of XML lends itself well to this formatting and allows for browsers to let users to naturally collapse individual tree elements. This feature would be particularly useful in debugging.

XML Advantages

3. Mixed content Support : XML is the capability of communication mixed content within same data payload. This mixed content is clearly differentiated with different markup tags.



The **XML DOM** defines a standard way for accessing and manipulating XML documents. It presents an XML document as a tree-structure.

- ❑ A standard object model for XML.
- ❑ A standard programming interface for XML.
- ❑ Platform and language-independent.



XML DOM

All XML elements can be accessed through the XML DOM.

```
<?xml version="1.0" encoding="UTF-8"?>
<bookstore>
  <book category="children">
    <title lang="en">Harry Potter</title>
    <author>J K. Rowling</author>
    <year>2005</year>
    <price>29.99</price>
  </book>
</bookstore>
```

This JavaScript code retrieves the text value of the first **<title>** element in an XML document:

```
let txt =
xmlDoc.getElementsByTagName("title")[0].childNodes[0].nodeValue;
```

XML DOM

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

```
<bookstore>
```

```
  <book category="children">
```

```
    <title lang="en">Everyday Italian</title>
```

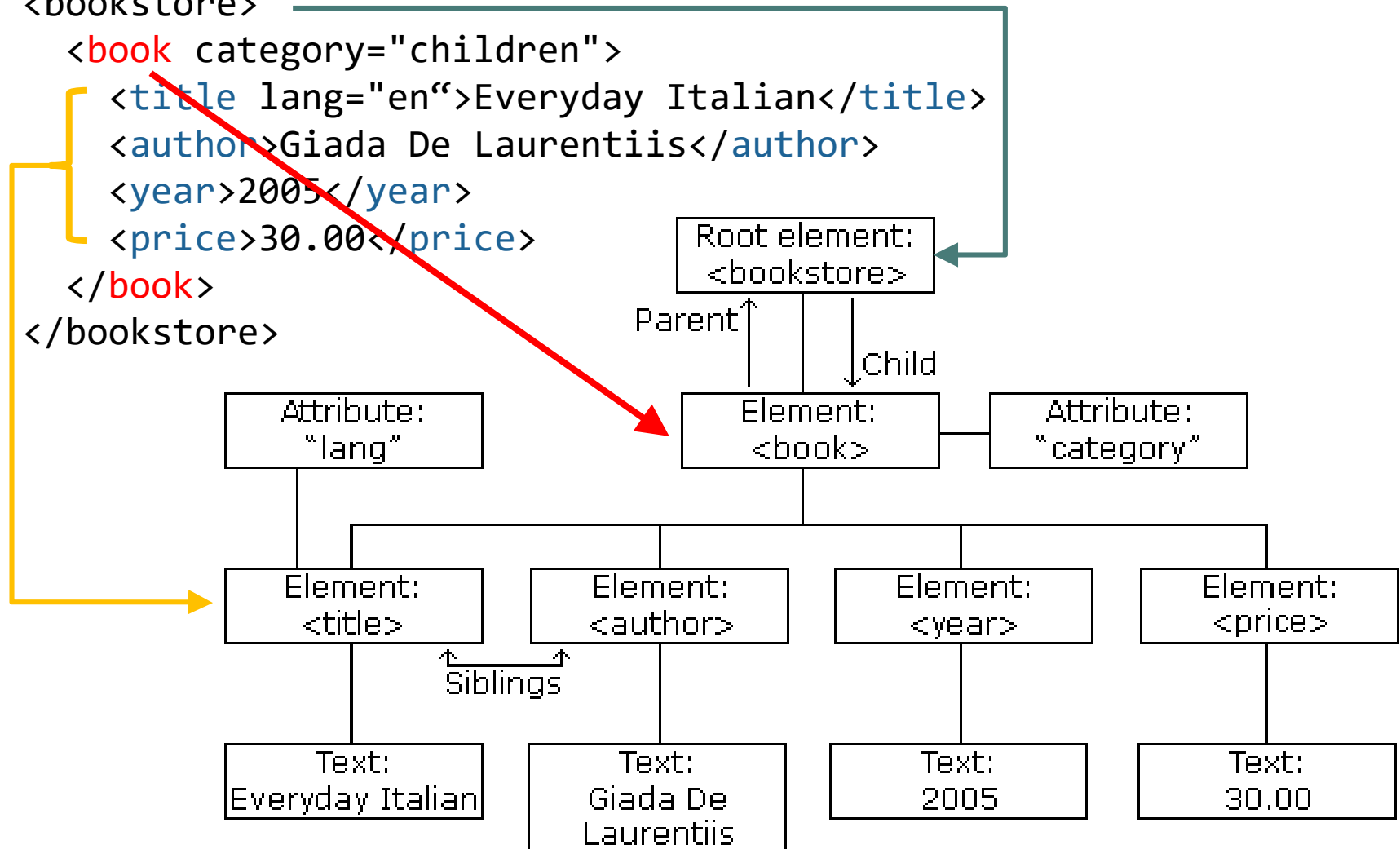
```
    <author>Giada De Laurentiis</author>
```

```
    <year>2005</year>
```

```
    <price>30.00</price>
```

```
  </book>
```

```
</bookstore>
```



More Information

- ❑ JavaScript Tutorial

<https://www.w3schools.com/js/default.asp>

- ❑ XML DOM Tutorial

https://www.w3schools.com/xml/dom_intro.asp

- ❑ XML DOM Tutorial

<https://www.tutorialspoint.com/dom/>