Assignment 3

Write a XML schema to describe and validate the XML data

An XML file is a file used to store data in the form of hierarchical elements. Data stored in XML files can be read by computer programs with the help of custom tags, which indicate the type of element. Since XML files are plain text documents, they are easy to create, store, transport, and interpret by computers and humans alike. This is why XML is one of the most commonly used languages on the internet. Many web-based software applications store information and send information to other apps in XML format.

XML Separates Data from Presentation

- XML does not carry any information about how to be displayed.
- The same XML data can be used in many different presentation scenarios.
- Because of this, with XML, there is a full separation between data and presentation.

In many HTML applications, XML is used to store or transport data, while HTML is used to format and display the same data.

XML Separates Data from HTML

- When displaying data in HTML, you should not have to edit the HTML file when the data changes.
- With XML, the data can be stored in separate XML files.
- With a few lines of JavaScript code, you can read an XML file and update the data content of any HTML page.

XML Does Not Use Predefined Tags

The XML language has no predefined tags.

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.

HTML works with predefined tags like <p>, <h1>, <table>, etc.

With XML, the author must define both the tags and the document structure.

XML is Extensible

Most XML applications will work as expected even if new data is added (or removed).

Imagine an application designed to display the original version of note.xml (<to> <from> <heading> <body>).

Then imagine a newer version of note.xml with added <date> and <hour> elements, and a removed <heading>.

Uses of XML

XML has a variety of uses for Web, e-business, and portable applications. The following are some of the many applications for which XML is useful:
Web publishing: XML allows you to create interactive pages, allows the customer to customize those pages, and makes creating e-commerce applications more intuitive. With XML, you store the data once and then render that content for different viewers or devices based on style sheet processing using an Extensible Style Language (XSL)/XSL Transformation (XSLT) processor.

Web searching and automating Web tasks: XML defines the type of information contained in a document, making it easier to return useful results when searching the Web:For example, using HTML to search for books authored by Tom Brown is likely to return instances of the term 'brown' outside of the context of author. Using XML restricts the search to the correct context (for example, the information contained in the <author> tag) and returns only the information that you want. By using XML, Web agents and robots (programs that automate Web searches or other tasks) are more efficient and produce more useful results.

General applications: XML provides a standard method to access information, making it easier for applications and devices of all kinds to use, store, transmit, and display data.

e-business applications: XML implementations make electronic data interchange (EDI) more accessible for information interchange, business-to-business transactions, and business-to-consumer transactions.

Metadata applications: XML makes it easier to express metadata in a portable, reusable format.

Pervasive computing: XML provides portable and structured information types for display on pervasive (wireless) computing devices such as personal digital assistants (PDAs), cellular phones, and others. For example, WML (Wireless Markup Language) and VoiceXML are currently evolving standards for describing visual and speech-driven wireless device interfaces.

Transaction Data

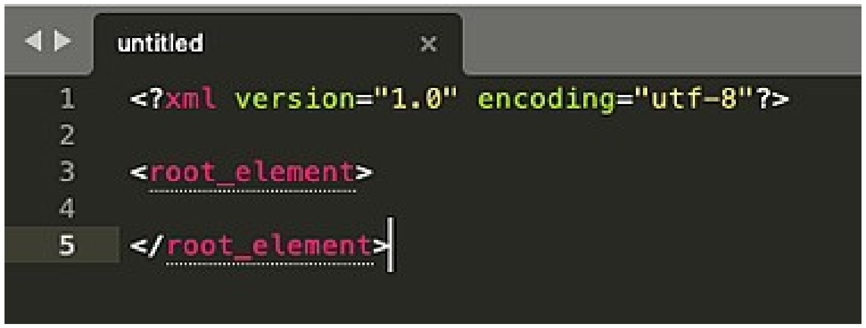
Thousands of XML formats exist, in many different industries, to describe day-to-day data transactions:

- Stocks and Shares
- Financial transactions
- Medical data
- Mathematical data
- Scientific measurements
- News information
- Weather services

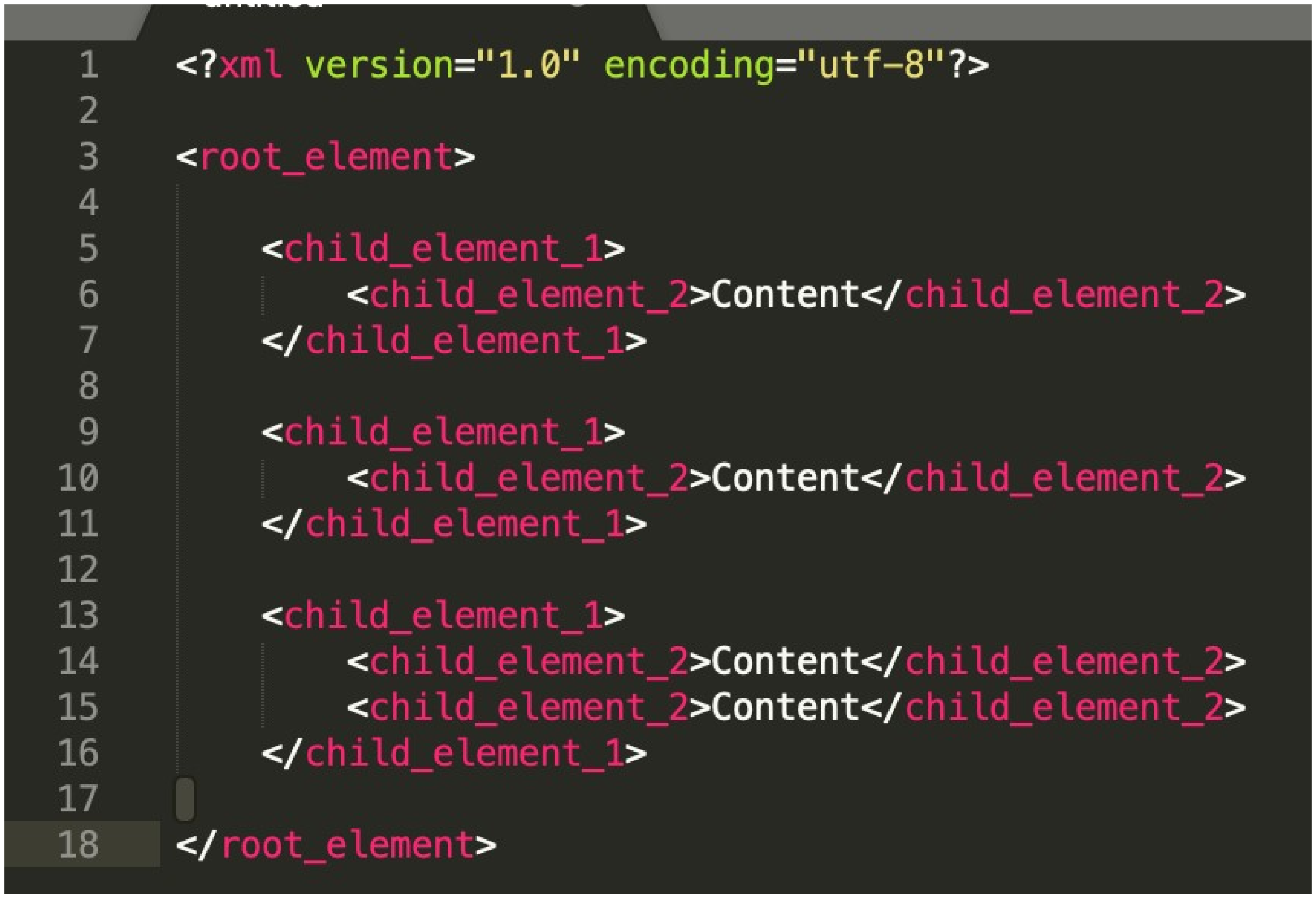
How to Create an XML File

- Open your text editor of choice.
- On the first line, write an XML declaration.
- Set your root element below the declaration.
- Add your child elements within the root element.
- Review your file for errors.
- Save your file with the .xml file extension.
- Test your file by opening it in the browser window.

Every XML file has one root element, which contains all other child elements. The root element is written below the declaration.



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

<root_element>

</root_element>
```

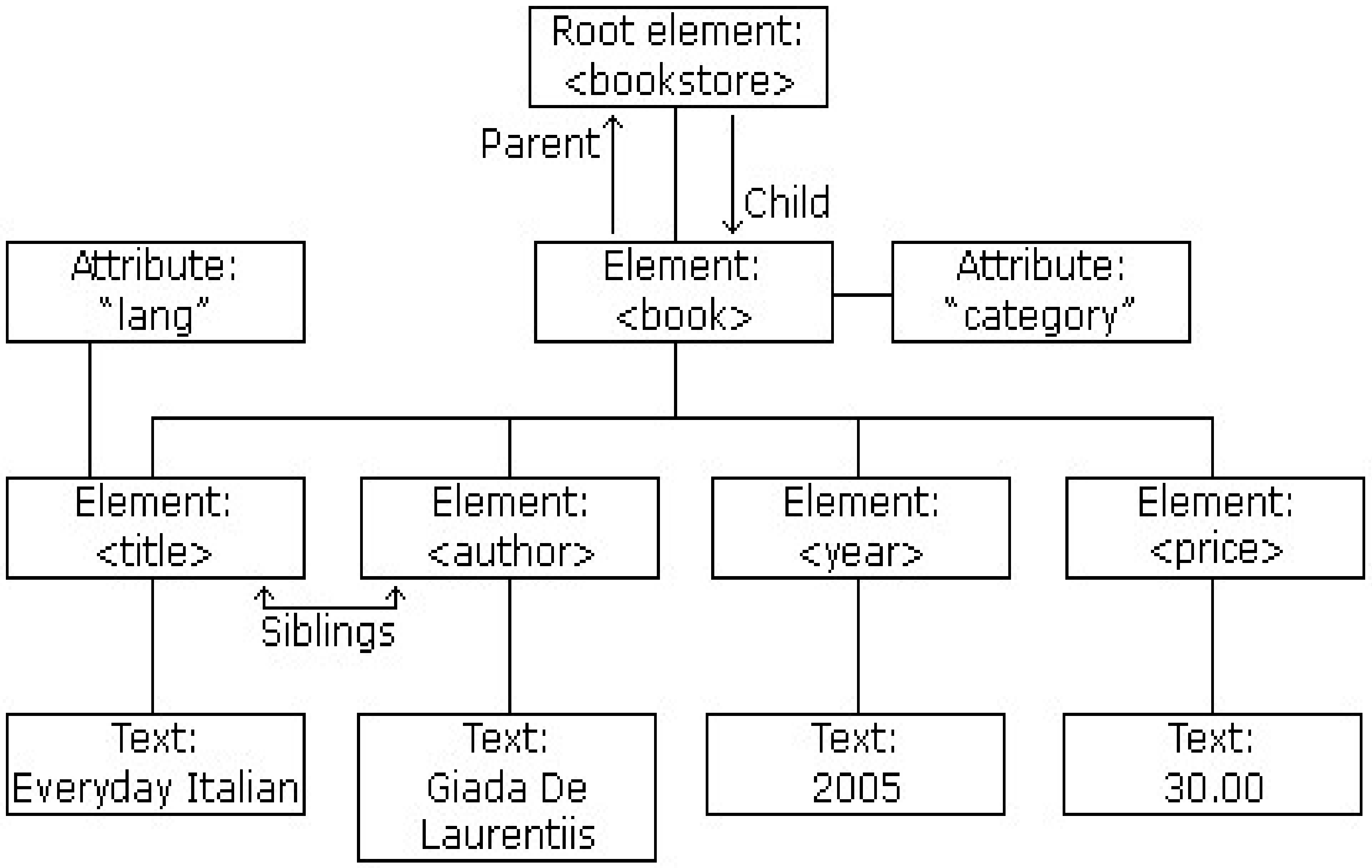


```xml
<?xml version="1.0" encoding="utf-8"?>

<root_element>

    <child_element_1>
        <child_element_2>Content</child_element_2>
    </child_element_1>

    <child_element_1>
        <child_element_2>Content</child_element_2>
    </child_element_1>

    <child_element_1>
        <child_element_2>Content</child_element_2>
        <child_element_2>Content</child_element_2>
    </child_element_1>

</root_element>
```

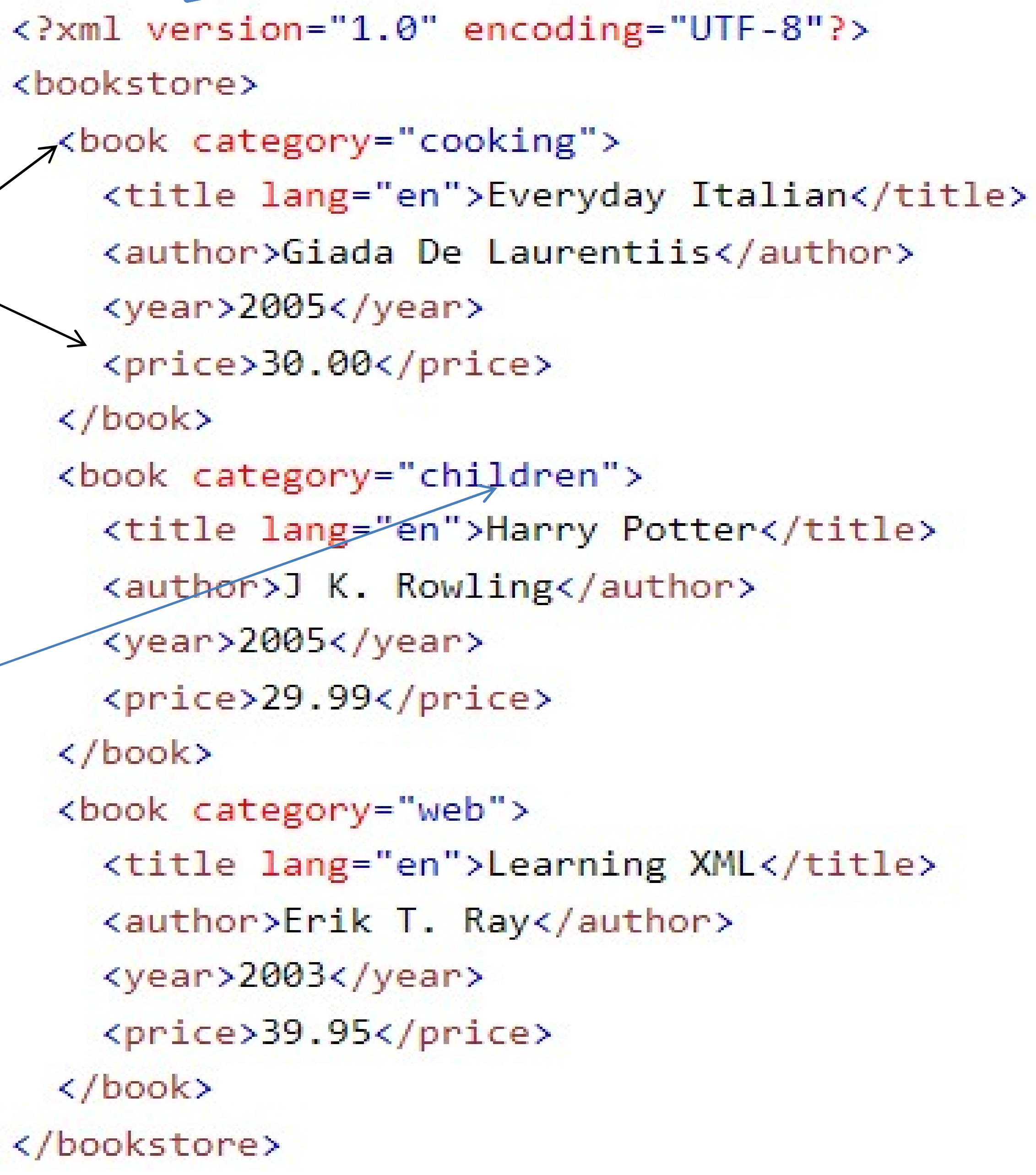
XML Tree

The XML Tree Structure

XML documents must contain one **root** element that is the **parent** of all other elements



The XML prolog is optional. If it exists, it must come first in the document.



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

An element can contain:
text
attributes
other elements
or a mix of the above

Attribute values must always be quoted. Either single or double quotes can be used.

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

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

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

This XML carries HTML table information:

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

This XML carries information about a table (a piece of furniture):

```
<table>
  <name>African Coffee Table</name>
  <width>80</width>
  <length>120</length>
</table>
```

XML Namespaces - The xmlns Attribute

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

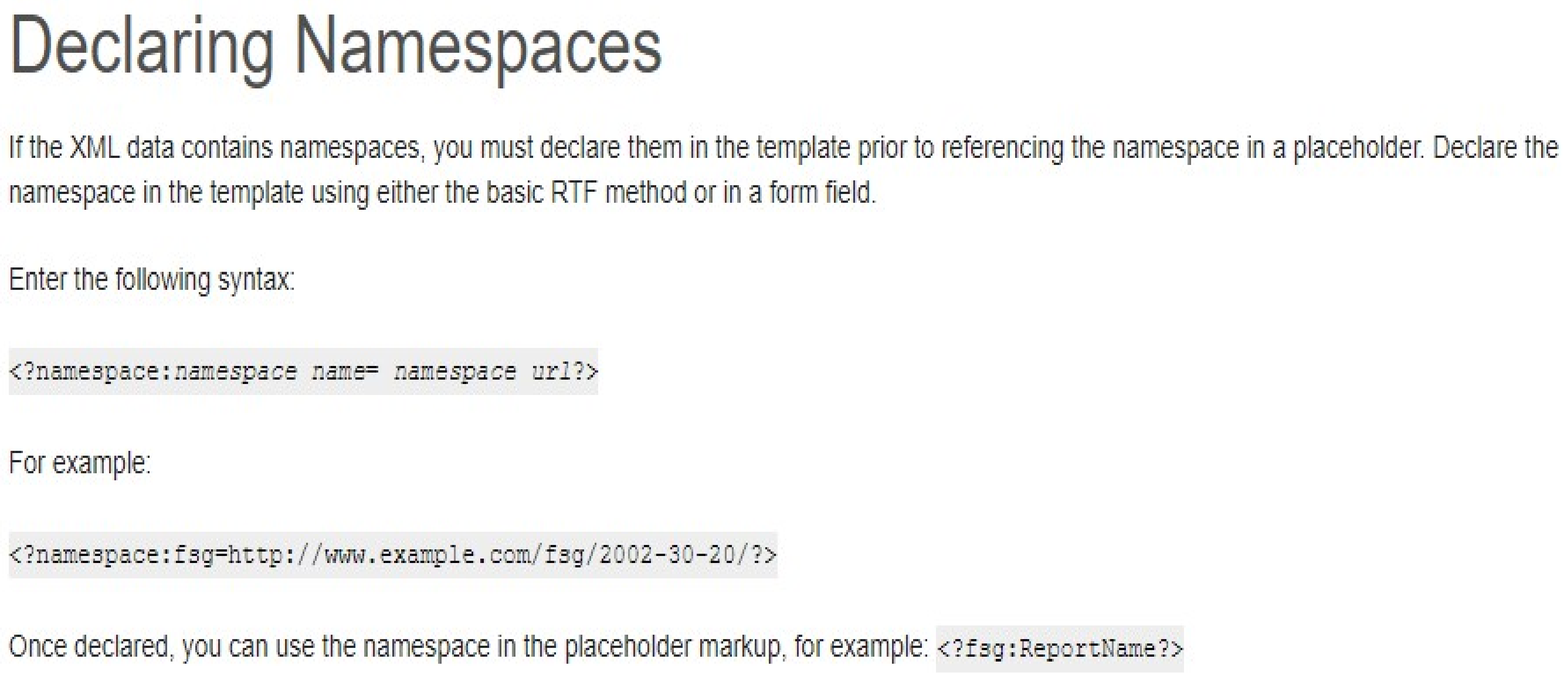
The namespace can be defined by an **xmlns** attribute in the start tag of an element.

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="https://www.w3schools.com/furniture">
  <f:name>African Coffee Table</f:name>
  <f:width>80</f:width>
  <f:length>120</f:length>
</f:table>

</root>
```

XML Namespaces

XML Namespaces provide a method to avoid element name conflicts.
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.



Declaring Namespaces

If the XML data contains namespaces, you must declare them in the template prior to referencing the namespace in a placeholder. Declare the namespace in the template using either the basic RTF method or in a form field.

Enter the following syntax:

```
<?namespace:namespace name= namespace url?>
```

For example:

```
<?namespace:fsg=http://www.example.com/fsg/2002-30-20/?>
```

Once declared, you can use the namespace in the placeholder markup, for example: `<?fsg:ReportName?>`

What is an 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)**.

It is used to describe and validate the structure and the content of XML data. XML schema defines the elements, attributes and data types. Schema element supports Namespaces. It is similar to a database schema that describes the data in a database.

The purpose of an XML Schema is to define the legal building blocks of an XML document:

- the elements and attributes that can appear in a document
- the number of (and order of) child elements
- data types for elements and attributes
- default and fixed values for elements and attributes

XML Schemas Support Data Types

One of the greatest strength of XML Schemas is the support for data types.

•It is easier to describe allowable document content
•It is easier to validate the correctness of data
•It is easier to define data facets (restrictions on data)
•It is easier to define data patterns (data formats)
•It is easier to convert data between different data types

XML Schemas use XML Syntax

Another great strength about XML Schemas is that they are written in XML.

•You don't have to learn a new language
•You can use your XML editor to edit your Schema files
•You can use your XML parser to parse your Schema files
•You can manipulate your Schema with the XML DOM
•You can transform your Schema with XSLT

XML Schemas are extensible, because they are written in XML.

XML Schemas Secure Data Communication

When sending data from a sender to a receiver, it is essential that both parts have the same "expectations" about the content.

With XML Schemas, the sender can describe the data in a way that the receiver will understand.

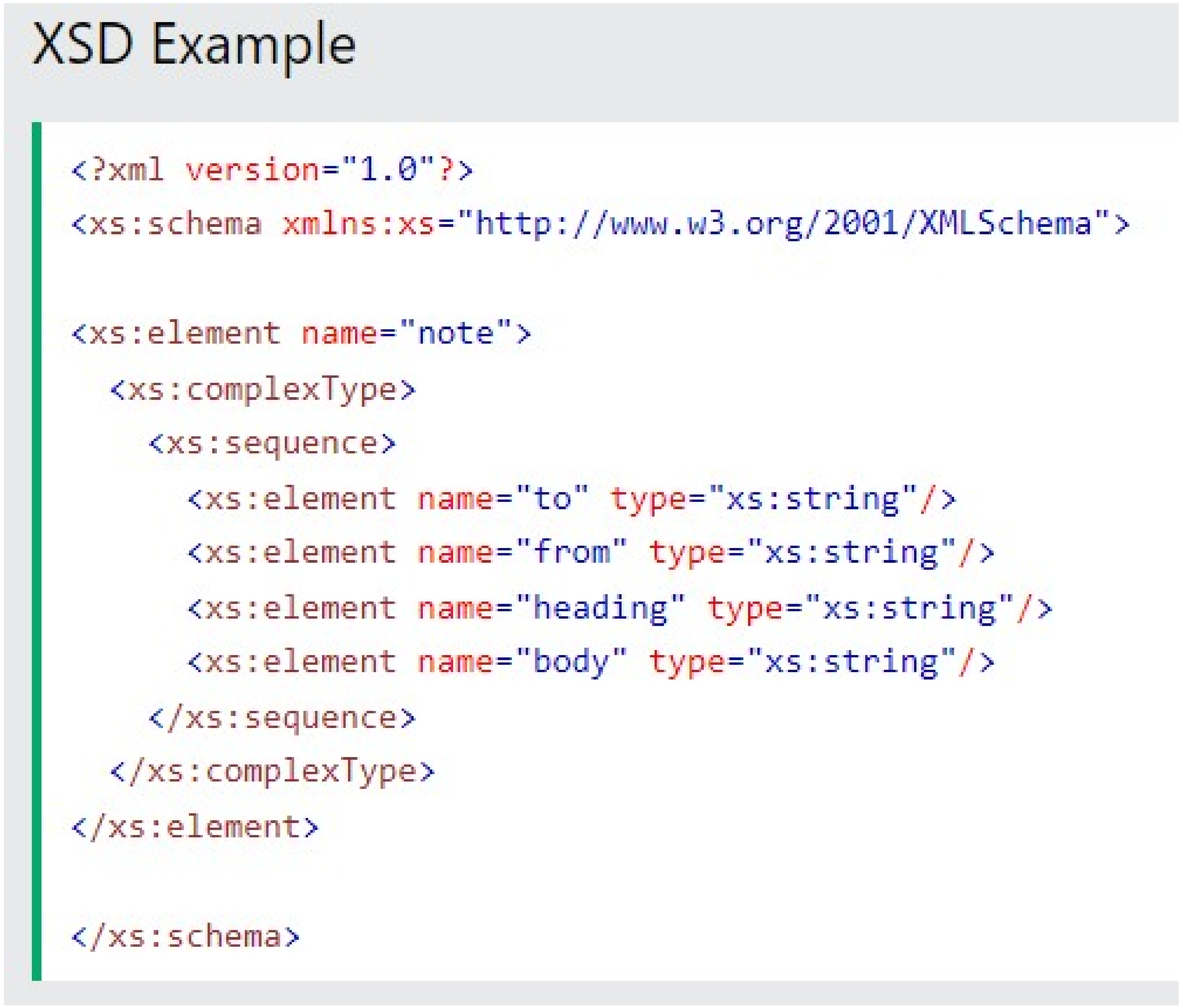
A date like: "03-11-2004" will, in some countries, be interpreted as 3.November and in other countries as 11.March.

However, an XML element with a data type like this:

```
<date type="date">2004-03-11</date>
```

ensures a mutual understanding of the content, because the XML data type "date" requires the format "YYYY-MM-DD".

Sample XSD	Sample XML
`<xs:element name="Customer_dob" type="xs:date"/>`	`<Customer_dob> 2000-01-12T12:13:14Z </Customer_dob>`
`<xs:element name="Customer_address" type="xs:string"/>`	`<Customer_address> 99 London Road </Customer_address>`
`<xs:element name="OrderID" type="xs:int"/>`	`<OrderID> 5756 </OrderID>`
`<xs:element name="Body" type="xs:string"/>`	`<Body> (a type can be defined as a string but not have any content; this is not true of all data types, however). </Body>`



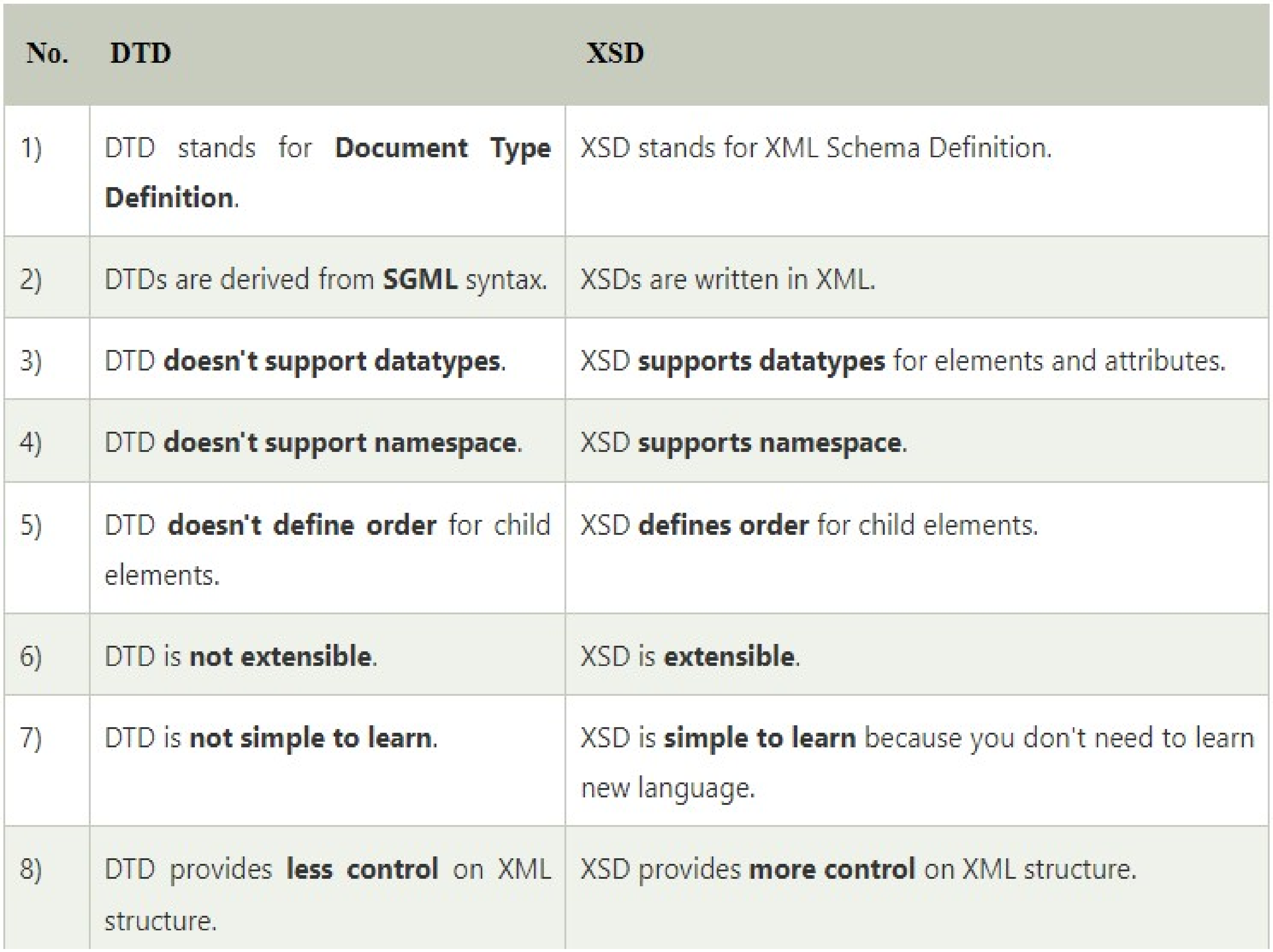
XSD Example

```
<?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>
```

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



No.	DTD	XSD
1)	DTD stands for **Document Type Definition**.	XSD stands for XML Schema Definition.
2)	DTDs are derived from **SGML** syntax.	XSDs are written in XML.
3)	DTD **doesn't support datatypes**.	XSD **supports datatypes** for elements and attributes.
4)	DTD **doesn't support namespace**.	XSD **supports namespace**.
5)	DTD **doesn't define order** for child elements.	XSD **defines order** for child elements.
6)	DTD is **not extensible**.	XSD is **extensible**.
7)	DTD is **not simple to learn**.	XSD is **simple to learn** because you don't need to learn new language.
8)	DTD provides **less control** on XML structure.	XSD provides **more control** on XML structure.

THANK YOU !