Module 4

XML Schema

Module Overview

In this module, you will learn about:

- XML Schema
- Exploring XML Schemas
- Working with Complex Types
- Working with Simple Types

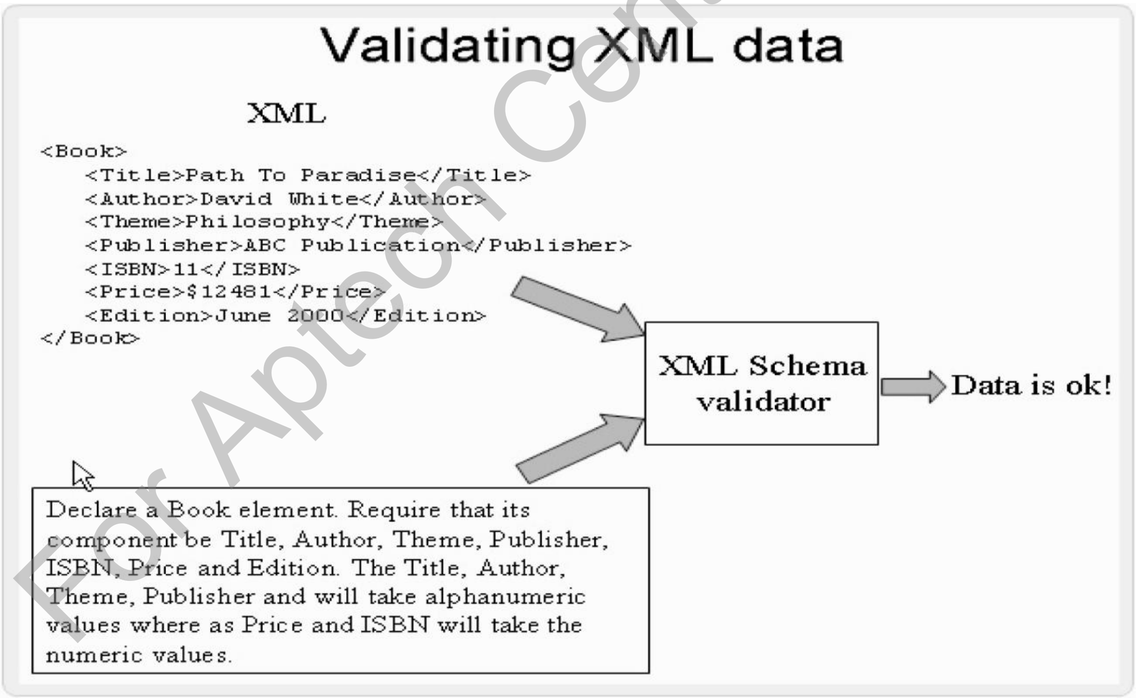
Lesson 1 – XML Schema

In this first lesson, **XML Schema**, you will learn to:

- Define and describe what is meant by schema.
- Identify the need for schema.
- Compare and differentiate the features of DTDs and schemas

XML Schema 1-2

- It defines the valid building blocks of an XML document.
- XML Schema language is referred as XML Schema Definition (XSD).
- It describes the data that is marked up, and also specifies the arrangement of tags and text.
- The dictionary defines a schema as "An outline or a model".



```
<Book>
    <Title>Path To Paradise</Title>
    <Author>David White</Author>
    <Theme>Philosophy</Theme>
    <Publisher>ABC Publication</Publisher>
    <ISBN>11</ ISBN>
    <Price>$12481</Price>
    <Edition>June 2000</Edition>
</Book>
```

Declare a Book element. Require that its component be Title, Author, Theme, Publisher, ISBN, Price and Edition. The Title, Author, Theme, Publisher and will take alphanumeric values where as Price and ISBN will take the numeric values.

XML Schema 2-2

Code Snippet

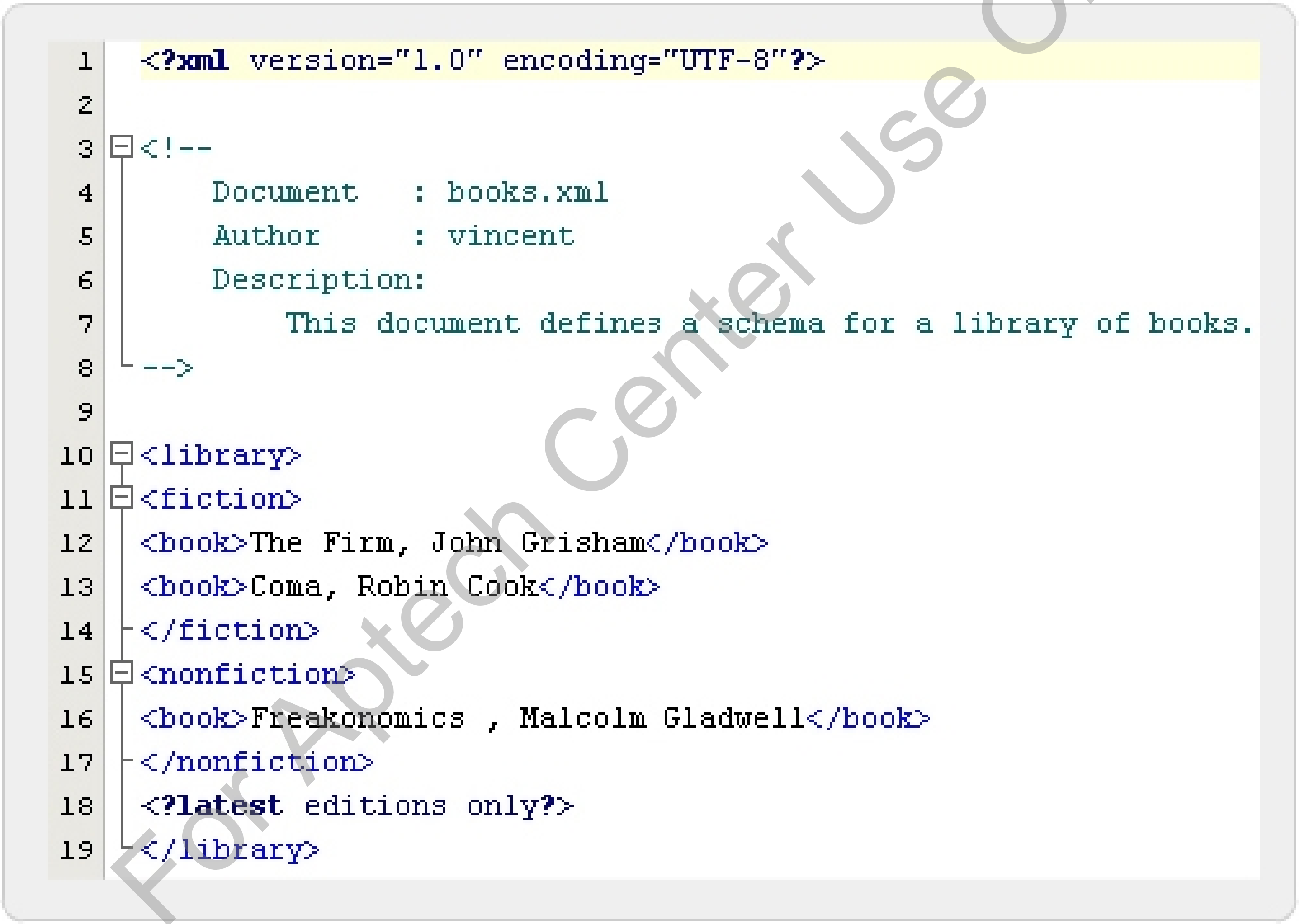
```
<Book>
  <Title>Path To Paradise</Title>
  <Author>David White</Author>
  <Theme>Philosophy</Theme>
  <Publisher>ABC Publication</Publisher>
  <ISBN>11</ISBN>
  <Price>$12481</Price>
  <Edition>June 2000</Edition>
</Book>
```

XML Schema Objectives 1-2

An XML Schema defines:

- Elements and attributes that can appear in a document
- Which elements are child elements
- The order and number of child elements
- Whether an element is empty or can include text
- Data types for elements and attributes
- Default and fixed values for elements and attributes

XML Schema Objectives 2-2



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

<!--
    Document   : books.xml
    Author     : vincent
    Description:
        This document defines a schema for a library of books.
-->

<library>
<fiction>
<book>The Firm, John Grisham</book>
<book>Coma, Robin Cook</book>
</fiction>
<nonfiction>
<book>Freakonomics , Malcolm Gladwell</book>
</nonfiction>
<?latest editions only?>
</library>
```

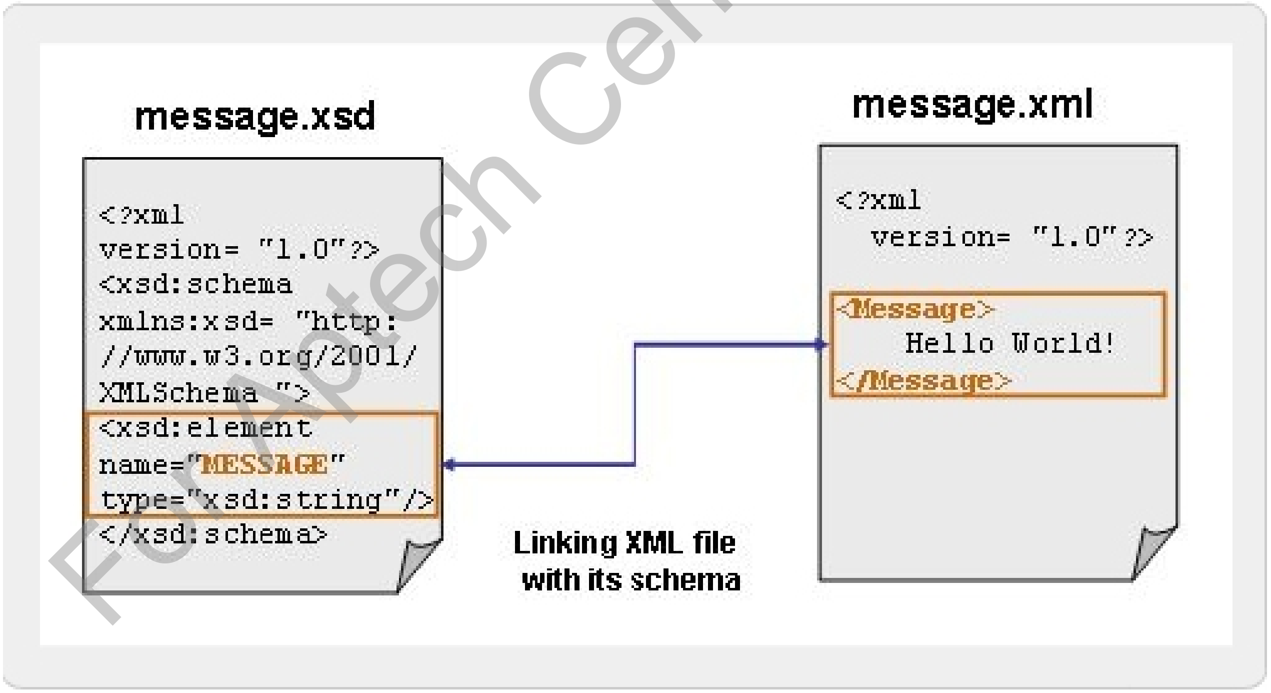
How to write an XML Schema? 1-2

XML File

- The XML document contains a single element, <Message>.

XSD File

- The file is saved with ".xsd" as the extension for storing schema documents



```
<?xml
version= "1.0"?>
<xsd:schema
xmlns:xsd= "http:
//www.w3.org/2001/
XMLSchema ">
<xsd:element
name="MESSAGE"
type="xsd:string"/>
</xsd:schema>
```

```
<?xml
  version= "1.0"?>
<Message>
    Hello World!
</Message>
```

How to write an XML Schema? 2-2

Code Snippet

```
XML File: message.xml
<?xml version= "1.0"?>
<Message>
    Hello World!
</Message>
```

```
XSD File: message.xsd
<?xml version= "1.0"?>
<xsd:schema xmlns:xsd= "http://www.w3.org/2001/XMLSchema
">
<xsd:element name= "MESSAGE " type= "xsd:string "/>
</xsd:schema>
```

What is an XML Schema? 1-3

- XML schemas allow Web applications to exchange XML data more robustly using the following range of new features:
  - Schemas support data types
  - Schemas are portable and efficient
  - Schemas secure data communication
  - Schemas are extensible
  - Schemas catch higher-level mistakes
  - Schemas support Namespace

Schemas support data types

- Support and ability to create the required datatypes has overcome the drawbacks of DTDs
- Easy to define and validate valid document content and data formats
- Easy to implement the restrictions on data

What is an XML Schema? 2-3

Schemas are portable and efficient

- Portable and efficient
- No need to learn any new language or any editor
- Similar XML parser can be used to parse the Schema files

Schemas secure data communication

- Sender can specify the data in a way that the receiver will understand

Schemas are extensible

- It is possible to reuse an existing schema to create another schema
- It is possible to create own data types derived from the standard types
- Support reference of multiple schemas in the same document

What is an XML Schema? 3-3

Schemas catch higher-level mistakes

- Catch higher-level mistakes that arise, such as a required field of information is missing or in a wrong format, or an element name is mis-spelled

Schemas support Namespace

- Support for XML Namespaces allows the programmer to validate documents that use markup from multiple namespaces
- Constructs can be re-used from schemas already defined in a different namespace

Comparing DTDs with Schemas 1-2

- Drawbacks of using DTDs are:
  - DTDs are written in a non-XML syntax
  - DTDs are not extensible
  - DTDs do not support namespaces
  - DTDS offer limited data typing

Sample External DTD File: program.dtd

```
<!ELEMENT program (comments, code)>
<!ELEMENT comments (#PCDATA)>
<!ELEMENT code (#PCDATA)>
```

Comparing DTDs with Schemas 2-2

Sample XML File with a reference to dtd: program.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE program SYSTEM "program.dtd">
<program>
    <comments>
        This is a simple Java Program. It will display the
message
        "Hello world!" on execution.
    </comments>
    <code>
        public static void main(String[] args)
        System.out.println("Hello World!"); // Display the
string.
    </code>
</program>
```

Advantages of XML Schemas over DTD 1-3

XML schema offers a range of new features:

- Richer data types
- Archetypes
- Attribute grouping
- Refinable archetypes

Advantages of XML Schemas over DTD 2-3

Sample schema File: mail.xsd

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.abc.com"
xmlns="http://www.abc.com"
elementFormDefault="qualified">
<xs:element name="mail">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="to" type="xs:string"/>
        <xs:element name="from" type="xs:string"/>
        <xs:element name="header" type="xs:string"/>
        <xs:element name="body" type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
</xs:element>
</xs:schema>
```

Advantages of XML Schemas over DTD 3-3

Sample XML File with a reference to schema: mail.xml

```
<?xml version="1.0"?>
<mail
xmlns="http://www.abc.com"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.abc.com mail.xsd">

<to>John</to>
<from>Jordan</from>
<header>Scheduler</header>
<body>3rd March Monday, 7:30 PM: board meeting!</body>
</mail>
```

Lesson 2 – Exploring XML Schemas

In this second lesson, **Exploring XML Schemas**, you will learn to:

- List the data types supported by schemas.
- Explain the XML Schema vocabulary.

Data Types Supported by Schema 1-6

- XML Schema describes a number of built-in data types, which can be used to specify and validate the intended data type of the content.

- It also allows a user to create a user-defined data type by extending the built-in data types using facets.

- XML Schema recommendation defines two sorts of data types:
  - Built-in data types
  - User-derived data types

Built-in data types

- Available to all XML Schema authors, and should be implemented by a conforming processor.

User-derived data types

- Defined in individual schema instances, and are particular to that schema.

Data Types Supported by Schema 2-6

string	▪ can contain characters, line feeds, carriage returns, and tab characters
boolean	▪ legal values for boolean data type are true and false ▪ true can be replaced by the numeric value 1 and false can be replaced by the value 0
numeric	▪ represents a numerical value ▪ includes numbers such as whole numbers, and real numbers
dateTime	▪ represents a particular time on a given date, written as a string
binary	▪ include graphic files, executable programs, or any other string of binary data
anyURI	▪ represents a file name or location of the file

Data Types Supported by Schema 3-6

Syntax: string

```
<xs:element name="element_name" type="xs:string"/>
```

Syntax: boolean

```
<xs:attribute name="attribute_name" type="xs:boolean"/>
```

Syntax: numeric

```
<xs:element name="element_name" type="xs:numeric"/>
```

Data Types Supported by Schema 4-6

Syntax: dateTime

```
<xs:element name="element_name" type="xs:dateTime"/>
```

Syntax: binary

```
<xs:element name="image_name" type="xs:hexBinary"/>
```

Syntax: anyURI

```
<xs:attribute name="image_name" type="xs:anyURI"/>
```

Data Types Supported by Schema 5-6

Code Snippet: string

```
<xs:element name="Customer" type="xs:string"/>
```

```
<Customer>John Smith</Customer>
```

Code Snippet: boolean

```
<xs:attribute name="Disabled" type="xs:boolean"/>
```

```
<Status Disabled="true">OFF</Status>
```

Code Snippet: numeric

```
<xs:element name="Price" type="xs:numeric"/>
```

```
<Price>500</Price>
```

Data Types Supported by Schema 6-6

Code Snippet: dateTime

```
<xs:element name="BeginAt" type="xs:dateTime"/>
```

```
<start>2001-05-10T12:35:40</start>
```

Code Snippet: binary

```
<xs:element name="Logo" type="xs:hexBinary"/>
```

Code Snippet: anyURI

```
<xs:attribute name="flower" type="xs:anyURI"/>
```

```
<image
flower="http://www.creativepictures.com/gallery/flower.gif
" />
```

Additional Data types 1-3

- Additional data types are derived from the basic built-in data types, which are called base type data types.
- The generated or derived data types, supported by the XML schema include:
  - integer
  - decimal
  - time

Integer

- Base type is numeric data type. Includes both positive and negative numbers. Used to specify a numeric value without a fractional component.

Decimal

- Represent exact fractional parts such as 3.26. Base type is numeric data type. Used to specify a numeric value.

Additional Data types 2-3

Time

- Base type is the dateTime data type. Default representation is 16:35:26. time data type is used to specify time. Specified as "hh:mm:ss".

Syntax: integer

```
<xs:element name="element_name" type="xs:integer"/>
```

Syntax: decimal

```
<xs:element name="element_name" type="xs:decimal"/>
```

Syntax: time

```
<xs:element name="element_name" type="xs:time"/>
```

Additional Data types 3-3

Code Snippet: integer

```
<xs:element name="Age" type="xs:integer"/>
```

```
<Age>999</Age>
```

Code Snippet: decimal

```
<xs:element name="Weight" type="xs:decimal"/>
```

```
<prize>+70.7860</prize>
```

Code Snippet: time

```
<xs:element name="BeginAt" type="xs:time"/>
```

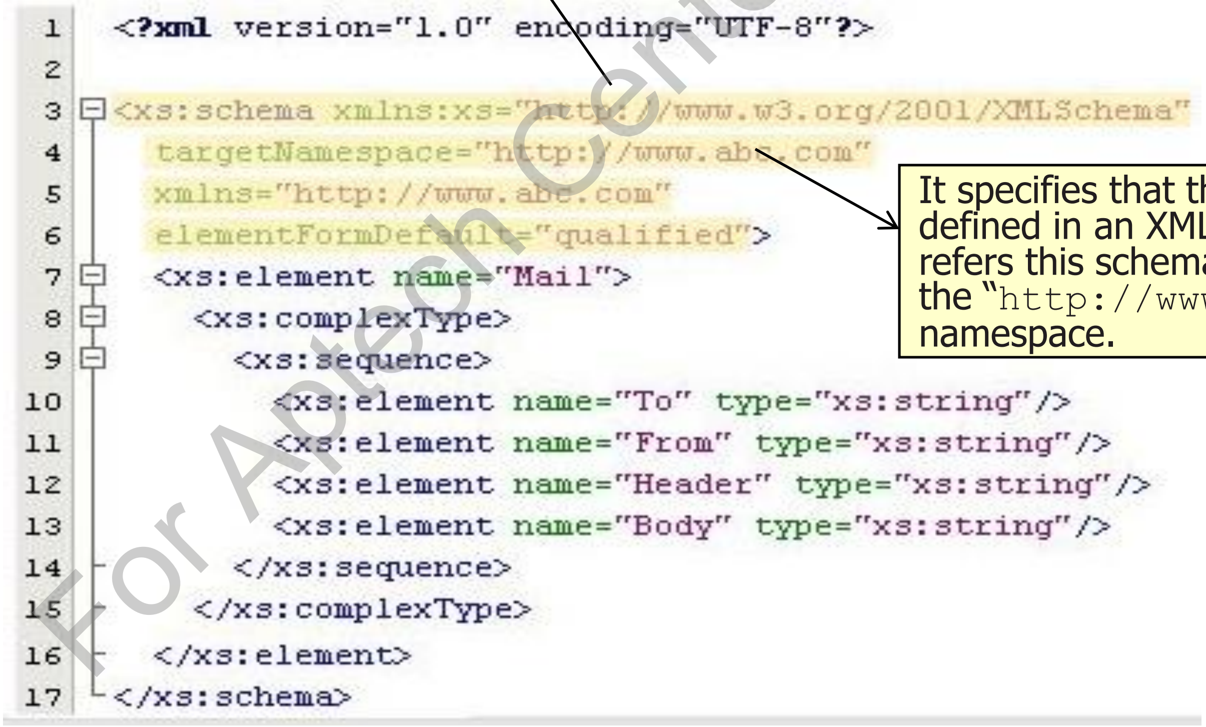
```
<BeginAt>09:30:10.5</BeginAt>
```

Schema Vocabulary 1-3

- Creating a schema using XML schema vocabulary is like creating any other XML document using a specialized vocabulary.

Every XML Schema starts with the root element `<schema>`.

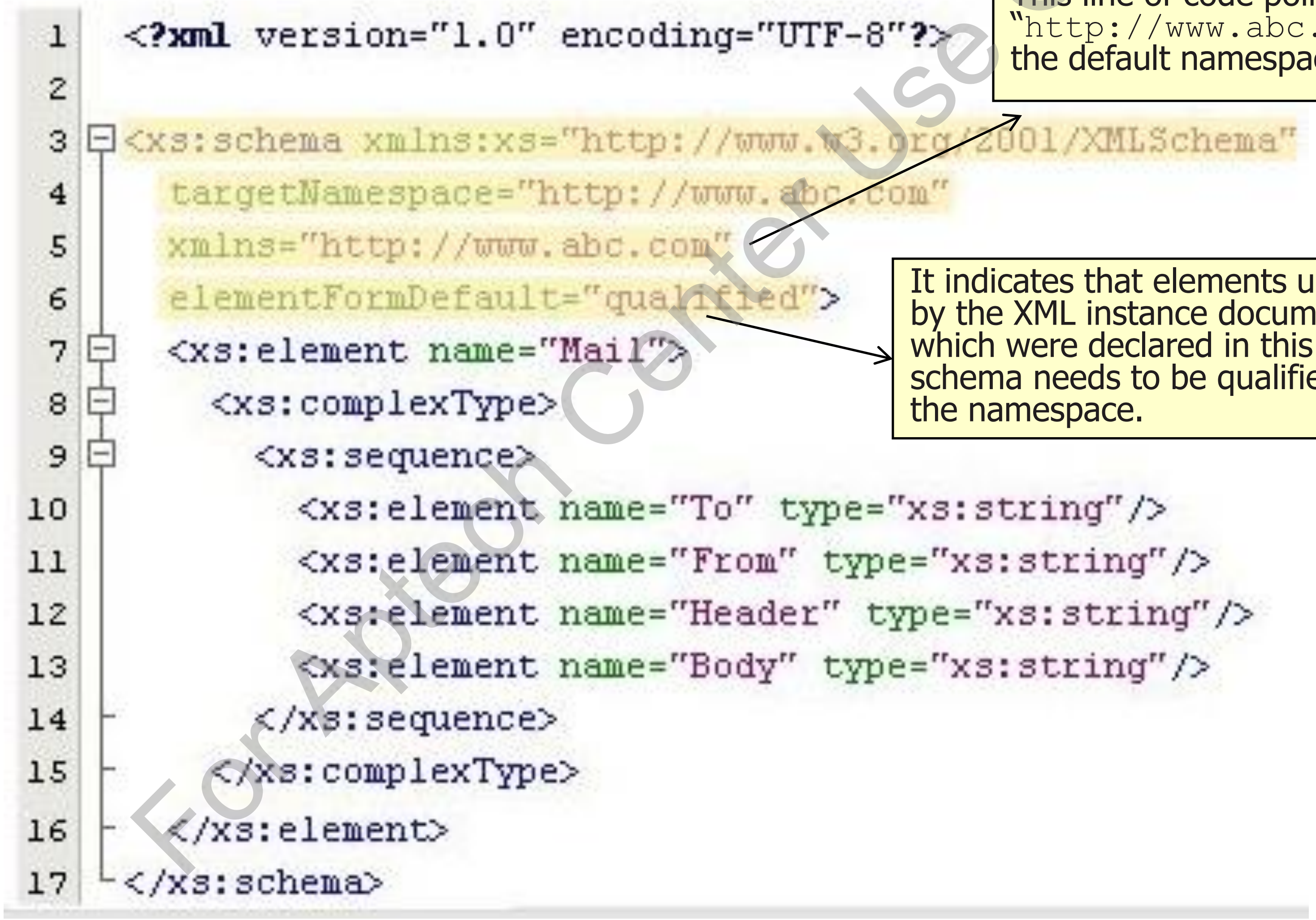
The code indicates that that the elements and data types used in the schema are derived from the "`http://www.w3.org/2001/XMLSchema`" namespace and prefixed with `xs`.



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

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://www.abc.com"
  xmlns="http://www.abc.com"
  elementFormDefault="qualified">
  <xs:element name="Mail">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="To" type="xs:string"/>
        <xs:element name="From" type="xs:string"/>
        <xs:element name="Header" type="xs:string"/>
        <xs:element name="Body" type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

It specifies that the elements defined in an XML document that refers this schema, come from the "`http://www.abc.com`" namespace.

Schema Vocabulary 2-3



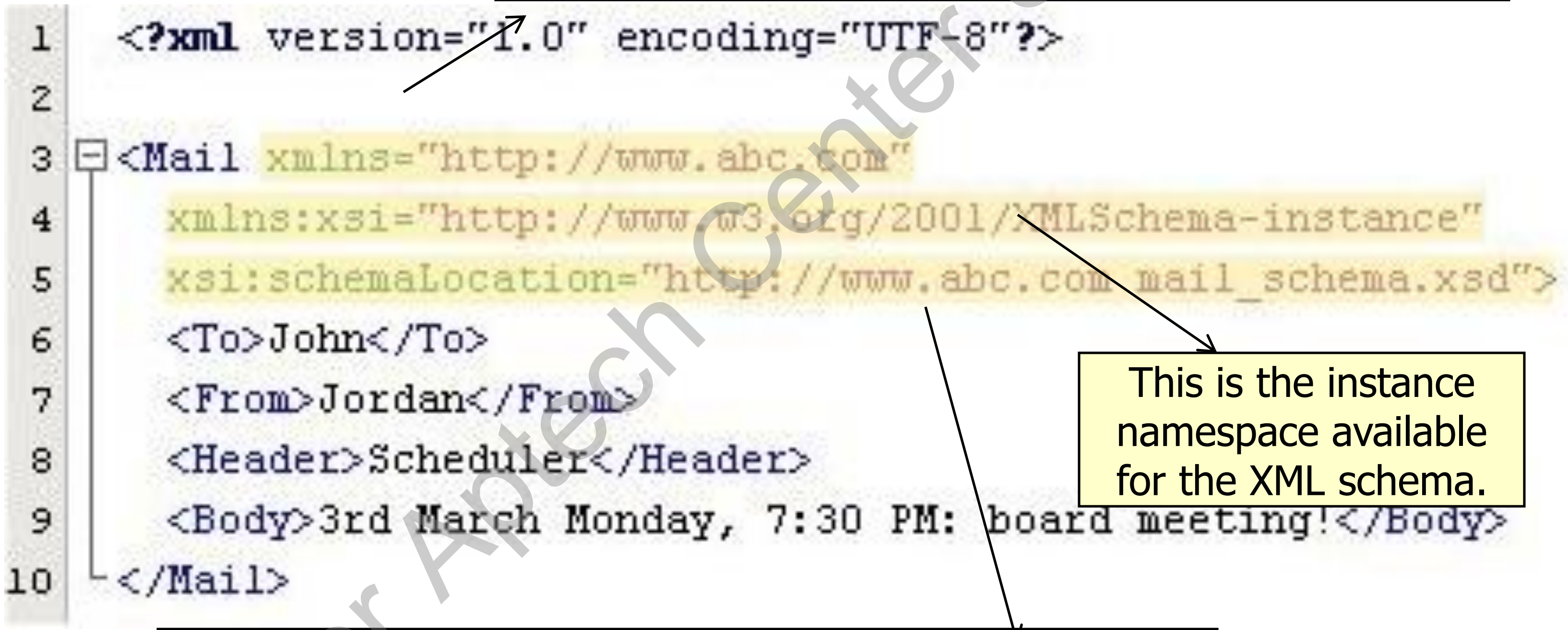
```
<?xml version="1.0" encoding="UTF-8"?>

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://www.abc.com"
  xmlns="http://www.abc.com"
  elementFormDefault="qualified">
  <xs:element name="Mail">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="To" type="xs:string"/>
        <xs:element name="From" type="xs:string"/>
        <xs:element name="Header" type="xs:string"/>
        <xs:element name="Body" type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

This line of code points to "`http://www.abc.com`" as the default namespace.

It indicates that elements used by the XML instance document which were declared in this schema needs to be qualified by the namespace.

Schema Vocabulary 3-3

It indicates the default namespace declaration. This declaration informs the schema-validator that all the elements used in this XML document are declared in the default (“http://www.abc.com”) namespace.



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

<Mail xmlns="http://www.abc.com"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.abc.com mail_schema.xsd">
    <To>John</To>
    <From>Jordan</From>
    <Header>Scheduler</Header>
    <Body>3rd March Monday, 7:30 PM: board meeting!</Body>
</Mail>
```

This is the instance namespace available for the XML schema.

This schemaLocation attribute has two values. The first value identifies the namespace. The second value is the location of the XML schema where it is stored.

Lesson 3 – Working with Complex Types

In this third lesson, **Working with Complex Types,** you will learn to:

- Describe complex type elements.
- Describe `minOccurs` and `maxOccurs`.
- Explain element content and mixed content.
- Describe how grouping can be done.

Complex Types 1-4

- A schema assigns a type to each element and attribute it declares.
- Elements with complex type may contain nested elements and have attributes.
- Only elements can contain complex types. Complex type elements have four variations. They are:
  - Empty Elements
  - Only Elements
  - Only Text
  - Mixed

Complex Types 2-4

Empty elements

- Empty elements optionally specify attributes types, but do not permit content.

```
<xs:element name="Books">
  <xs:complexType>
    <xs:attributename="BookCode"
     type="xs:positiveInteger"/>
  </xs:complexType>
</xs:element>
```

Only Elements

- These elements can only contain elements and do not contain attributes.

```
<xs:element name="Books">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="ISBN" type="xs:string"/>
      <xs:element name="Price" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Complex Types 3-4

Only Text

- These elements can only contain text and optionally may or may not have attributes.

```
<xs:complexType name="Books">
    <xs:simpleContent>
        <xs:extension base="xs:string">
            <xs:attribute name="BookCode" type="xs:
positiveInteger"/>
        </xs:extension>
    </xs:simpleContent>
</xs:complexType>
```

Complex Types 4-4

Mixed

- These are elements that can contain text content as well as sub-elements within the element. They may or may not have attributes.

```
<xs:element name="Books">
  <xs:complexType mixed="true">
    <xs:sequence>
      <xs:element name="BookName" type="xs:string"/>
      <xs:element name="ISBN" type="xs:positiveInteger"/>
      <xs:element name="Price" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

A complex element can be defined in two different ways:

- By directly naming the element
- By using the name and type attribute of the complex type

Defining Complex Types 1-2

- **By directly naming the element**

Code Snippet

```
<xs:element name=”Student”>
 <xs:complexType>
  <xs:sequence>
   <xs:element name=”FirstName” type=”xs:string”/>
   <xs:element name=”MiddleName” type=”xs:string”/>
   <xs:element name=”LastName” type=”xs:string”/>
  </xs:sequence>
 </xs:complexType>
</xs:element>
```

Defining Complex Types 2-2

- **By using the name and type attribute of the complex type**

Code Snippet

```
<xs:element name="Student" type="PersonInfo"/>
<xs:complexType name="personinfo">
 <xs:sequence>
  <xs:element name="FirstName" type="xs:string"/>
  <xs:element name="LastName" type="xs:string"/>
 </xs:sequence>
</xs:complexType>
```

minOccurs and maxOccurs 1-3

- **minOccurs**
  - It specify the minimum number of occurrences of the element in an XML document.
- **maxOccurs**
  - It specify the maximum number of occurrences of the element in an XML document.

minOccurs and maxOccurs 2-3

Code Snippet

```
…
<xs:element name= "Books">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="ISBN" type="xs:string"/>
   <xs:element name="Quantity" type="xs:string"
maxOccurs="100" minOccurs="0"/>
  </xs:sequence>
 </xs:complexType>
</xs:element>
…
```

minOccurs and maxOccurs 3-3

minOccur	MaxOccur	Number of times an element can occur
0	1	0 or 1
1	1	1
0	*	Infinite
1	*	At least once
>0	*	At least minOccurs times
>maxOccurs	>0	0
Any value	<minOccurs	0

Element Content 1-2

Code Snippet

```
// Books.xml

<?xml version="1.0"?>
<Books xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation= "Books.xsd">
<Title>A cup of tea</Title>
<Author>
<Name>Dennis Compton</Name>
</Author>
<Author>
<Name>George Ford</Name>
</Author>
<Publisher>
<Name>Orange</Name>
</Publisher>
</Books>
```

Element Content 2-2

Code Snippet

```
// Books.xsd

<?xml version="1.0"?>
 <xs:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xs:element name= "Books" type= "BookType">
   <xs:complexType name= "AuthorType">
    <xs:sequence>
   <xs:element name= "Name" type="xs:string"/>
    </xs:sequence>
   </xs:complexType>
    <xs:complexType name= "PublisherType">
     <xs:sequence>
      <xs:element name= "Name" type= "xs:string"/>
       </xs:sequence>
      </xs:complexType>
     <xs:complexType name= "BookType">
      <xs:sequence>
        <xs:element name= "Title" type= "xs:string"/>
        <xs:element name= "Author" type="ComposerType"
maxOccurs= "unbounded"/>
        <xs:element name= "Publisher" type="PublisherType"
minOccurs="0" maxOccurs= "unbounded"/
```

Mixed Content

Code Snippet

```
// Book.xml
<Books>
Apocalypse written by<Author>Mary Jane</Author>
is of Genre<Category>Fiction</Type>.
</Books>
```

```
// Book.xsd
<xs:element name="Books">
 <xs:complexType mixed="true">
  <xs:sequence>
   <xs:element name="Author" type="xs:string"/>
   <xs:element name="Category" type="xs:string"/>
   </xs:sequence>
 </xs:complexType>
</xs:element>
```

Grouping Constructs 1-3

xs:all

- This grouping construct requires that each element in the group must occur at most once

Code Snippet

```
<xs:element name= “Books”>
 <xs:complexType>
  <xs:all>
   <xs:element name=”Name” type=”xs:string” minOccurs= “1”
   maxOccurs= “1”/>
   <xs:element name=”ISBN” type=”xs:string” minOccurs= “1”
   maxOccurs= “1”/>
   <xs:element name=”items” type=”Items” minOccurs= “1” />
  </xs:all>
 </xs:complexType>
</xs:element>
```

Grouping Constructs 2-3

`xs:sequence`

- It specifies each member of the sequence to appear in the same order

Code Snippet

```
<xs:element name=”Books”>
<xs:complexType>
<xs:sequence>
<xs:element name=”Name” type=”xs:string” />
<xs:element name=”ISBN” type=” xs:string 〃 />
<xs:element name=”Price” type=” xs:string 〃 />
</xs:sequence>
</xs:complexType>
</xs:element>
```

Grouping Constructs 3-3

`xs:choice`

- It will allow only one of the choices to appear instead of requiring all the elements to be present

Code Snippet

```
<xs:complexType name="AdressInfo">
  <xs:group>
    <xs:choice>
      <xs:element name="Address" type="USAddress" />
      <xs:element name="Address" type="UKAddress" />
      <xs:element name="Address" type="FranceAddress" />
    </xs:choice>
  </xs:group>
```

Lesson 4 – Working with Simple Types

In this last lesson, **Working with Simple Types,** you will learn to:

- Describe simple types.
- List and describe the data types used with simple types.
- Explain restrictions and facets.
- Identify the usage of attributes.

Defining a Simple Type Element

- They are used to form the textual data and specify the type of data allowed within attributes and elements.

Syntax

```
<xs:element name="XXXX" type="YYYY"/>
```

Code Snippet

```
Book.xml: XML Elements
<Book_name>The Da vinci code</Book_name>
<TotalNoOfPages>360</TotalNoOfPages>
<Author_name>Dan Brown</Author_name>
```

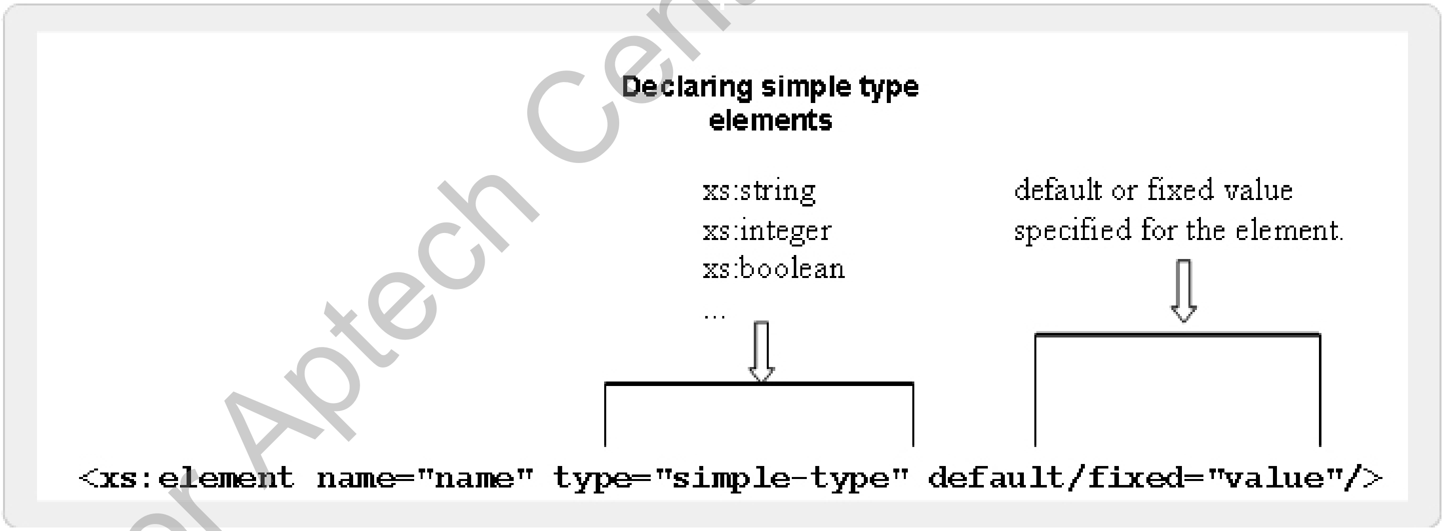
```
Book.xsd: Corresponding simple element definitions
<xs:element name="Book_name" type="xs:string" />
<xs:element name="TotalNoOfPages"
type="xs:integer"/>
<xs:element name="Author_name" type="xs:string"/>
```

Data types used with Simple types

- Built-in simple type
- User-defined simple type

Built-in Simple types 1-2

- There are several built-in simple types, such as integer, date, float and string.
- It can contain a default value or a fixed value.


Declaring simple type elements
xs:string
xs:integer
xs:boolean
...
default or fixed value specified for the element.

```
<xs:element name="name" type="simple-type" default/fixed="value"/>
```

Built-in Simple types 2-2

Syntax

```
<xs:element name=”XXXX” type=”YYYY” default=”ZZZZ”/>
```

Code Snippet

```
<xs:element name=”AccountType” type=”xs:string”
fixed=”Savings”/>
<xs:element name=”BalanceAmount” type=”xs:integer”
default=”5000”/>
```

User-defined Simple types 1-2

- Custom user defined datatype can be created using the <simpleType> definition.

Syntax

```
<xs:simpleType name="name of the simpleType">
 <xs:restriction base="built-in data type">
  <xs:constraint="set constraint to limit the
content"/>
 </xsd:restriction>
</xsd:simpleType>
```

User-defined Simple types 2-2

Code Snippet 1

```
<xs:simpleType name=”AngleMeasure”>
 <xs:restriction base=”xs:integer”>
  <xs:minInclusive value=”0”/>
  <xs:maxInclusive value=”360”/>
 </xs:restriction></xs:simpleType>
```

Code Snippet 2

```
<xs:simpleType name=”triangle”>
 <xs:restriction base=”xsd:string”>
  <xs:enumeration value=”isosceles”/>
  <xs:enumeration value=”right-angled”/>
  <xs:enumeration value=”equilateral”/>
 </xs:restriction>
</xs:simpleType>
```

Restrictions 1-2

- It can be specified for the simpleType elements.
- They are declared using the <restriction> declaration.

Syntax

```
<restriction base="name of the simpleType you are
deriving from">
```

In this `<restriction>` declaration, the `base data` type can be specified using the `base` attribute.

Code Snippet

```
<xs:simpleType name="Age">
  <xs:restriction base="xs:integer">
    ...
    ...
  </xs:restriction>
</xs:simpleType>
```

Restrictions 2-2

Syntax

```
<xs:simpleType name= “name”>
    <xs:restriction base= “xs:source”>
        <xs:facet value= “value”/>
        <xs:facet value= “value”/>
        …
    </xs:restriction>
</xs:simpleType>
```

Code Snippet

```
<xs:simpleType name=”triangle”>
  <xs:restriction base=”xs:string”>
    <xs:enumeration value=”isosceles”/>
    <xs:enumeration value=”right-angled”/>
    <xs:enumeration value=”equilateral”/>
  </xs:restriction>
</xs:simpleType>
```

Facets 1-2

- They are used to restrict the set or range of values a datatype can contain. There are 12 facet elements declared using a common syntax.

Facet	Description
`minExclusive`	Specifies the minimum value for the type that excludes the value provided.
`minInclusive`	Specifies the minimum value for the type that includes the value provided.
`maxExclusive`	Specifies the maximum value for the type that excludes the value provided.
`maxInclusive`	Specifies the maximum value for the type that includes the value provided.
`totalDigits`	Specifies the total number of digits in a numeric type.
`fractionDigits`	Specifies the number of fractional digits in a numeric type.
`length`	Specifies the number of items in a list type or the number of characters in a string type.
`minLength`	Specifies the minimum number of items in a list type or the minimum number of characters in a string type.
`maxLength`	Specifies the maximum number of items in a list type or the maximum number of characters in a string type.
`enumeration`	Specifies an allowable value in an enumerated list.
`whiteSpace`	Specifies how whitespace should be treated within the type.
`pattern`	Restricts string types.

Facets 2-2

Syntax

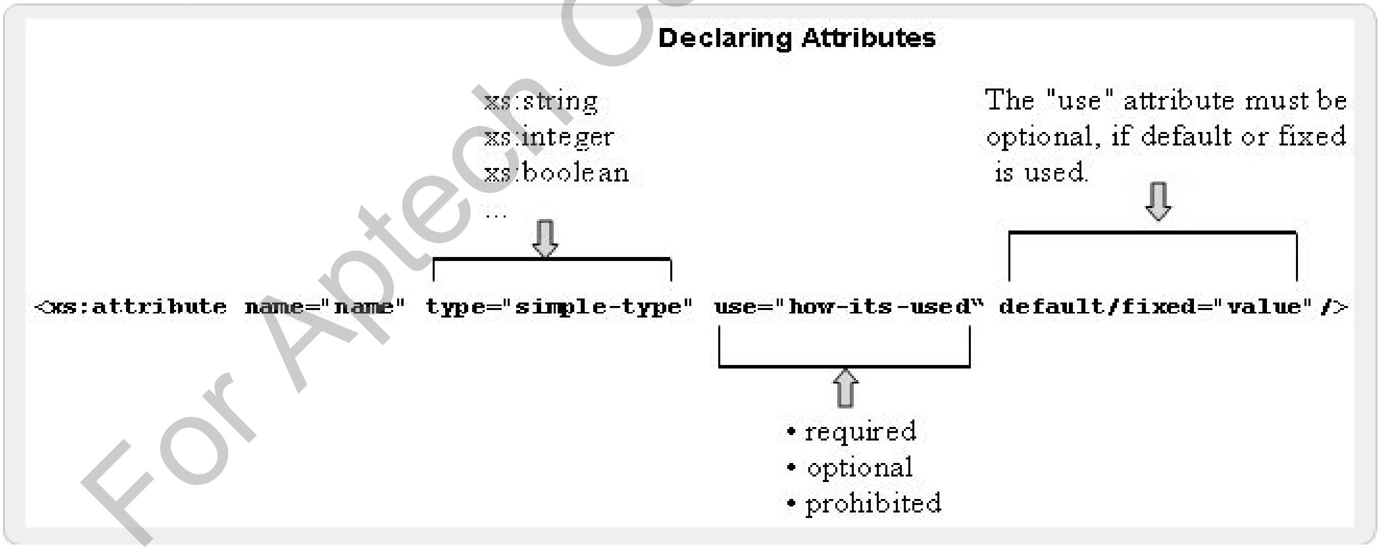
```
<xs:simpleType name= “name”>
  <xs:restriction base= “xs:source”>
    <xs:facet value= “value”/>
    <xs:facet value= “value”/>
    …
  </xs:restriction>
</xs:simpleType>
```

Code Snippet

```
<xs:simpleType name=”triangle”>
  <xs:restriction base=”xs:string”>
    <xs:enumeration value=”isosceles”/>
    <xs:enumeration value=”right-angled”/>
    <xs:enumeration value=”equilateral”/>
  </xs:restriction>
</xs:simpleType>
```

Attributes 1-3

- Default
- Fixed
- Optional
- Prohibited
- Required



```
<xs:attribute name="name" type="simple-type" use="how-its-used" default/fixed="value" />
```

Attributes 2-3

Syntax

```
<xs:attribute name="Attribute_name"
type="Attribute_datatype"/>
```

where,

`Attribute_name` is the name of the attribute.

`Attribute_datatype` specifies the data type of the attribute. There are lot of built in data types in XML schema such as string, decimal, integer, boolean, date, and time.

Attributes 3-3

Code Snippet

```
....
<xs:complexType name= “SingerType”>
 <xs:sequence>
  <xs:element name= “Name”>
   <xs:complexType>
    <xs:all>
     <xs:element name= “FirstName” type=”xs:string”/>
     <xs:element name= “LastName” type=”xs:string”/>
    </xs:all>
   </xs:complexType>
  </xs:element>
 </xs:sequence>
<xs:attribute name= ”age” type=”xs:positiveInteger”
use= “optional/>
</xs:complexType>
....
```

Summary 1-3

- **XML Schema**
  - An XML Schema is an XML based alternative to DTDs, which describes the structure of an XML document.
  - An XML Schema can define elements, attributes, child elements and the possible values that can appear in a document.
  - Schemas overcome the limitations of DTDs and allow Web applications to exchange XML data robustly, without relying on adhoc validation tools.
- **Exploring XML Schemas**
  - XML schema offers built-in and user defined data types.
  - It supports built-in data types like string, boolean, number, dateTime, binary, and uri.
  - It also supports integer, decimal, time, and user-defined data types.

Summary 2-3

- **Working with Complex Types**
  - Elements with complex type may contain nested elements and have attributes.
  - A complex element can be defined by directly naming the element and by using the name and the type attribute of the complex type.
  - `minOccurs` and `maxOccurs` specify the minimum and maximum number of occurrences of the element in an XML document respectively.
  - Element content in an XML document contains only XML elements and mixed content contains text mixed with elements.
  - The grouping constructs in XML schema specify the order of XML elements.

Summary 3-3

- **Working with Simple Types**
  - The elements of simple type describe the content and data type of the element rather than its structure.
  - The simple type can have built-in and user defined data types.
  - Simple type definition takes one of the two values, default or fixed as per the requirement.
  - The user-defined data type can be derived from a built-in type or an existing simple type.
  - Use of restrictions and facets restricts its content to user prescribed values.
  - Schema supports usage of attributes in elements, which is declared with a simple type definition.