XML Schema Definition

Objectives

At the end of this module, you will be able to:

- Describe the use of XML Schema and limitations of DTDs
- Describe structure of XML Schema
- Create Schema Definition Files
- Define attributes in XSD
- Apply restrictions on values and set of values

XML Schema

XML Schema

An XML-based language alternative to DTDs

Describes the structure of an XML document

The W3C XML Schema language is also referred to as XML Schema Definition or XSD

XML documents that comply to an XML schema are known as instances of that schema

- A well-formed XML document follows all the syntax rules of XML, but it may not necessarily adhere to any particular schema. So, an XML document can be well formed without being valid, but it cannot be valid unless it is well formed.

Limitations of DTDs

- **DTDs are a weak specification language**
  - You cannot put *any* restrictions on element contents
  - You have little control over ordering of elements
    - For example: Difficult to specify that all child elements must occur, but may be in any order
  - There are only ten data types for attribute type values

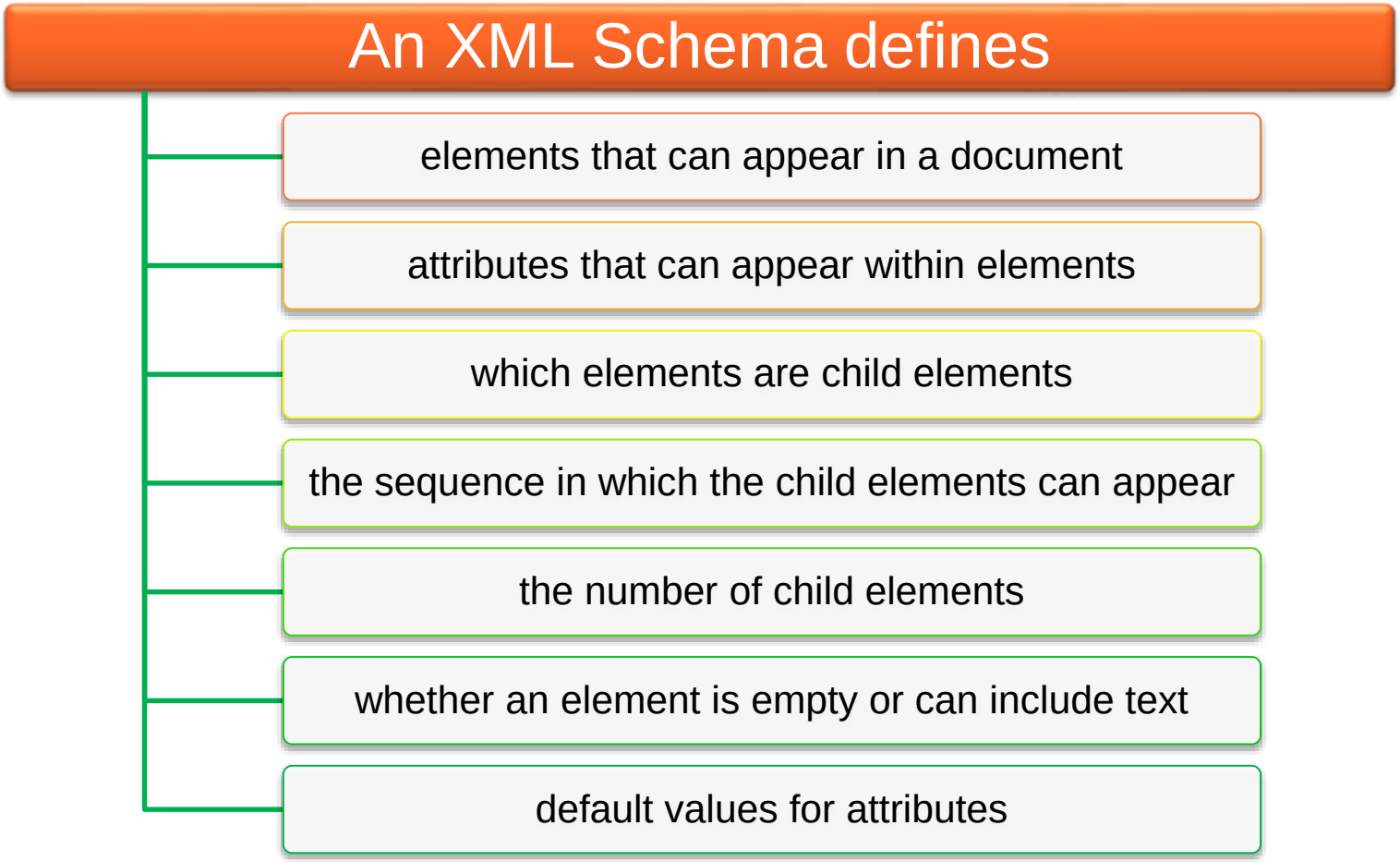
- **DTDs are not written in XML**
  - For validation, you need separate parsers for XML and DTD

Features of XML Schema

- XML Schema presents enhancements over DTDs:
  - Support for data types
    - 37+ built-in data types
    - Allows to create your own data types
  - It is written in XML
  - Can define the child elements to occur in any order
  - Gives you much more control over structure and content

Defining XML Schema

The purpose of XML Schema is to define the structure of an XML document just like a DTD



Example – a DTD to an XSD

- Consider a simple DTD called song.dtd and let us write the schema song.xsd

```
<!ELEMENT song (title, category, artist)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT category (#PCDATA)>
<!ELEMENT artist (#PCDATA)>
```

- The file extension of XSD document is .xsd

XSD - Example

song.xsd

```
<xsd:schema
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://www.music.org/album"
  elementFormDefault="qualified">
  <xsd:element name=“song">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="title" type="xsd:string"/>
        <xsd:element name=“category” type="xsd:string"/>
        <xsd:element name=“artist" type="xsd:string"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

Understanding XSD Example

- Every XSD document has a root element called <schema>
- Schema heavily uses namespaces, hence namespace declaration is done at root level itself

song.xsd

```
<xsd:schema
1   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
2   targetNamespace="http://www.music.org/album"
3   elementFormDefault="qualified">
    ……
</xsd:schema>
```

1. Namespace of XML Schema Language
2. Namespace of the vocabulary defined in this schema
3. All XML elements must be qualified (use a namespace)

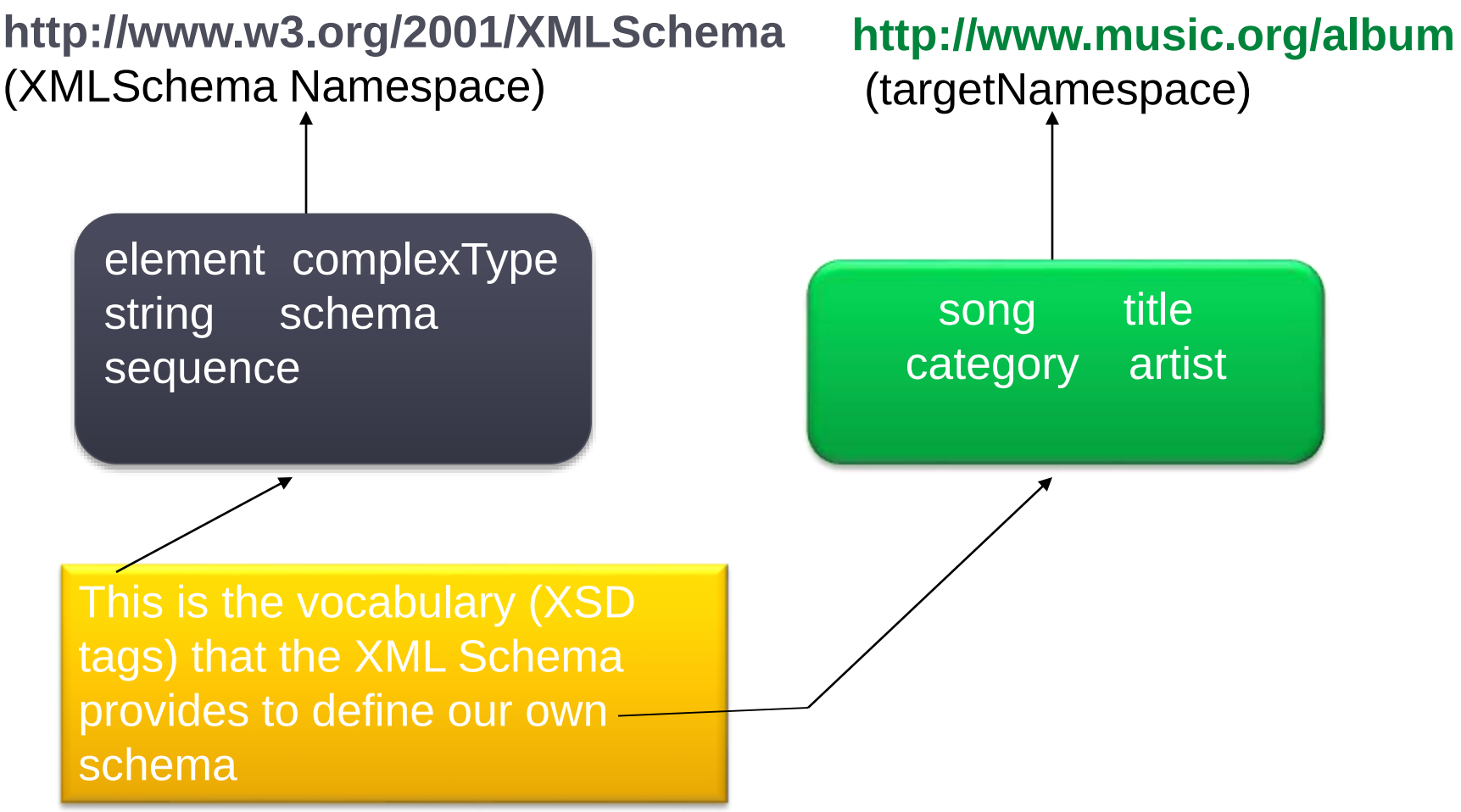
Understanding XSD Example

- The <schema> element may have attributes (namespaces)
- XML Schema elements are referred with the *xsd* prefix

  xmlns:xsd="http://www.w3.org/2001/XMLSchema" -This is necessary to specify since our XSD tags are defined here

- The elements and data types such as **schema**, **element**, **complexType**, **sequence**, **string**, etc that are used to construct schemas come from the **http://www.w3.org/2001/XMLSchema** namespace
- **targetNamespace=http://www.music.org/album** indicates that the elements defined by this schema such as: **song**, **title**, **category**, and **artist** are to go in the namespace specified by targetNamespace

Mapping of vocabulary and schema



XML Document mapped to Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<song
   xmlns="http://www.music.org/album"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.music.org/album song.xsd">
   <title>New Divide</title>
   <category>Pop</category>
   <artist>Linkin Park</artist>
</song>
```

The namespace to use

Informs parser about location of the schema when it defines a namespace

The location of XML schema to use for that namespace

Referencing a schema in an XML instance document

- To refer to a DTD in an XML document, the reference goes *before* the root element:

```
<?xml version="1.0"?>
<!DOCTYPE rootElement SYSTEM "url">
<rootElement> ... </rootElement>
```

- To refer to an XSD in an XML document, the reference goes *in* the root element:

```
<?xml version="1.0"?>
<rootElement
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance“
        (The XML Schema Instance reference is required)
   xsi:SchemaLocation="url   file">
          (This is where your XML Schema definition can be found)

        ...
</rootElement>
```

Multiple Levels of Checking

song.xml

song.xsd

XMLSchema.xsd
(schema-for-schemas)

Validate that xml document conforms to rules described in song.xsd

Validate that song.xsd is a valid schema document, i.e., it conforms to rules described in schema-for-schemas

Components of a Schema

- All schema definitions contain two types of elements: simple and complex

- A “simple” element is one that contains only text
  - Text can be simple data such as numbers, strings, or dates
  - Various restrictions may be applied to simple data
  - Cannot contain any child elements
  - Cannot have attributes

- A “complex” element is one that contains child elements
  - May contain attributes
  - May be empty, or it may contain text

Defining a simple element

- A simple element is defined as
-

```
<xsd:element   name="aaa"   type="bbb" />
```

- where:
  - aaa is the name of the element
  - bbb is the data type of the element
  - Common built-in types in XML schema

xsd:boolean	**xsd:integer**
xsd:date	**xsd:string**
xsd:decimal	**xsd:time**

- Elements may have a default value or a fixed value specified whose values are assigned automatically
  - default="*default value*" if no other value is specified
  - fixed="*value*" i.e., no other value may be specified

Defining simple element - Examples

- XML code for simple elements

```
<firstname>Anny</firstname>
 <age>19</age>
 <birthdate>1990-05-15</birthdate>
```

- XSD code for simple element definitions

```
<xsd:element name="firstname" type="xsd:string"/>
 <xsd:element name="age" type="xsd:integer"/>
 <xsd:element name="birthdate" type="xsd:date"/>
```

- Example: the default value is "ABC Inc":

```
<xsd:element name="company" type="xsd:string" default="ABC Inc"/>
```

Defining a complex element

- Use the element complexType when declaring elements with child elements

A complex element is defined as:

```
<xs:element   name="name">
    <xs:complexType>
   ….
      .... information of complex type element
    </xs:complexType>
</xs:element>
```

Defining a complex element - Example

- Example:

```
<xsd:element name="song">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="title" type="xsd:string"/>
        <xsd:element name="category" type="xsd:string"/>
        <xsd:element name="artist" type="xsd:string"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
```

- <xsd:sequence> means elements must occur in this order
- The <maxOccurs> indicator specifies the maximum number of times an element can occur.
- The <minOccurs> indicator specifies the minimum number of times an element can occur.

Defining Attributes

- Attributes are always declared as simple types
- An attribute is defined as
  <xsd:attribute name="aaa" type="bbb" />
  where:
  - aaa is the name of the attribute
  - bbb is the data type of the attribute
  - Common built-in types in XML schema

xsd:boolean	**xsd:integer**
xsd:date	**xsd:string**
xsd:decimal	**xsd:time**

- Example:
  - An XML element with an attribute:
    <book price="500">Mastering XML</book>
  - The corresponding attribute definition:
    <xsd:attribute name="price" type="xsd:integer"/>

Default and Fixed values for Attributes

Attributes may have a default value or a fixed value specified

- default=“default value” means no other value is specified
- Example:

```
<xs:attribute name=“publisher" type="xs:string" default=“ABC“/>
```

- fixed=“value” means no other value may be specified
- Example:

```
<xs:attribute name=“publisher" type="xs:string" fixed=“ABC“/>
```

Optional and Required Attributes

- Attributes are optional by default
- To specify that the attribute is required, use the "use" attribute
- Example:

```
<xs:attribute name=“title" type="xs:string" use="required“ />
```

The “use” attribute

- The use attribute specifies whether the attribute is required or optional
- If an attribute is optional, it specifies whether its value is fixed or has a default
- Examples:

```
<xs:attribute name=“title" type="xs:string" use="required“/>
<xsd:attribute name=“emailid" type="xsd:string" use="optional“ />
```

The “value” attribute

- The value attribute contains a value if you need to specify one

- Examples:

```
<xsd:attribute name="year" type="xsd:int" use="fixed" value="2007“ />

<xsd:attribute name="year" type="xsd:int" use="default" value="2007“ />
```

Restrictions or Facets

- You can put restrictions to define acceptable values for XML elements or attributes

- Restrictions on Values - Example

```
<xsd:element  name=“marks">
    <xsd:restriction base="xsd:integer">
        <xsd:minInclusive value="0">
        <xsd:maxInclusive value="100">
    </xsd:restriction>
</xsd:element>
```

- Here, an element called “marks" is defined with a restriction
  - The value of marks cannot be < 0 or > 100

Restrictions on a set of values

- You can use enumeration to restrict the value to be one of a fixed set of acceptable values

- For example: An element “sport” is defined with a restriction. The only acceptable values are: Tennis, Golf, Badminton

```
<xs:element name=“sport">
<xs:simpleType>
 <xs:restriction base="xs:string">
  <xs:enumeration value=“Tennis"/>
  <xs:enumeration value="Golf"/>
  <xs:enumeration value=“Badminton"/>
 </xs:restriction>
</xs:simpleType>
</xs:element>
```

Elements containing only other elements

- Example of XML element "employee“ which contains only other elements

```
<employee>
   <name>Anny</name>
   <address>Park Street, Bangalore</address>
</employee>
```

- Create a named complexType “person” and use that type

```
<xs:complexType name="persontype">
      <xs:sequence>
      <xs:element name="name" type="xs:string"/>
      <xs:element name=“address" type="xs:string"/>
      </xs:sequence>
</xs:complexType>
<xs:element name="employee"  type="persontype“ />
```

Elements containing only other elements (Contd.).

- In this method, several elements can refer to the same complex type
- For Example:

```
<xs:complexType name="persontype">
 <xs:sequence>
  <xs:element name="name" type="xs:string"/>
  <xs:element name="address" type="xs:string"/>
 </xs:sequence>
</xs:complexType>

  <xs:element name="employee" type="persontype"/>
  <xs:element name="customer" type="persontype"/>
  <xs:element name="faculty" type="persontype"/>
```

Using “ref” attribute

- You can design your schema by defining all elements and attributes first
- You can then refer to them using ref attribute
- Example:

```
<xsd:element name="name" type="xsd:string"/>
<xsd:element name="address" type="xsd:string"/>

<xsd:element name="employee">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="name"/>
        <xsd:element ref="address"/>
      </xsd:sequence>
    </xsd:complexType>
</xsd:element>
```

Definition of simple elements

Definition of complex element

Elements with mixed content

- Mixed complex type elements can contain attributes, elements, and text
- Add mixed="true" to the xs:complexType element
- An XML element, “message", that contains both text and other elements:

```
<message>
  Dear<name>Anny</name>,
  Please attend a training on<training>XML</training>
  scheduled on <tdate>2009-08-17</tdate>.
</message>
```

Elements with mixed content (Contd.).

- The following schema declares the “message" element

```
<xs:element name="message">
  <xs:complexType mixed="true“>
    <xs:sequence>
      <xs:element name="name" type="xs:string"/>
      <xs:element name="training" type="xs:string"/>
      <xs:element name="tdate" type="xs:date"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Summary

In this module, you were able to

- Describe the use of XML Schema and limitations of DTDs
- Describe structure of XML Schema
- Create Schema Definition Files
- Define attributes in XSD
- Apply restrictions on values and set of values

Thank You