

XML Schemas Reference Manual

Roger L. Costello
XML Technologies Course

Purpose

- The previous two Powerpoint documents showed how to write schemas. Many (but not all) of the features of XML Schemas was presented.
- The purpose of this document is to serve as a complete reference manual. For each XML Schema component (attribute, element, complexType, simpleType, group, attributeGroup, and annotation) I summarize the capabilities of the components, and explain each capability. As always, there will be lots of examples. I hope that you find this useful!

Items used to declare attributes

Here are the items that you use in declaring attributes:

name: use this to declare what will be the name of the attribute in instance documents.

ref: use this to refer-to an attribute declaration.

type: use this to declare the datatype of the attribute's value in instance documents.

default: use this to give the attribute a default value.

fixed: use this to give the attribute a fixed (constant) value.

use: use this to specify whether the attribute is `optional`, `required`, or `prohibited`. If a default or fixed value is specified in the attribute declaration then **use** must have the value `optional`. The default value of **use** is `optional`.

Cont. -->

Items used to declare attributes

Continued ...

- id:** use this to associate a unique identifier to the attribute component. This id has no representation in the instance document. It is purely internal to the schema. The type of this value must be `xsd:ID`.
- form:** use this to override **attributeFormDefault** (which is specified on the schema element). There are two possible values - `qualified` or `unqualified`. If **form** is assigned the value `qualified` then the attribute must be namespace-qualified in instance documents.

global versus local attributes

Attributes may be declared locally or globally:

1. Ways to create local attribute declarations:
 - 1.1 Inline an attribute declaration within a complexType.
 - 1.2 Inline an attribute declaration within an attributeGroup
2. Global attribute declarations are immediate children of <schema>

```
<xsd:complexType name="Airplane">  
  <xsd:attribute name="altitude" type="xsd:integer"/> 1.1  
  <xsd:attribute ref="speed"/>  
  <xsd:attributeGroup ref="physicalCharacteristics"/>  
</xsd:complexType>  
<xsd:attribute name="speed" type="xsd:decimal"/> 2.  
<xsd:attributeGroup name="physicalCharacteristics">  
  <xsd:attribute name="weight" type="xsd:decimal"/>  
  <xsd:attribute name="wing-span" type="xsd:decimal"/> 1.2  
</xsd:attributeGroup>
```

attributeFormDefault versus form

< **schema** > has an optional attribute, **attributeFormDefault**.

Example: <xsd:schema ...**attributeFormDefault**="qualified" ...>

The legal values for **attributeFormDefault** are *qualified* and *unqualified*.

The default value of **attributeFormDefault** is *unqualified*.

If **attributeFormDefault**="unqualified" then local attributes must not be namespace-qualified in instance documents.

If **attributeFormDefault**="qualified" then all attributes must be namespace-qualified in instance documents.

An attribute declaration may override **attributeFormDefault** with **form**:

<xsd:**attribute** ... **form**="qualified"/> This attribute must be namespace-qualified in instance documents.

When must attributes be namespace-qualified?

```
<image i:src="http://www.xfront.com/BestPractices.html"/>
```

An attribute must be namespace-qualified in instance documents if:

- when the attribute (e.g., src) was declared you specified **form="qualified"**, or
- the attribute was declared globally, or
- there was no value specified for **form** when the attribute was declared, but the schema has: **<schema ... attributeFormDefault="qualified"...>**

20 ways to declare attributes

- On the following slides I have identified 20 different ways to declare attributes.
- With each way to declare an attribute I have provided an example.
 - The folder `example37` contains all the examples.

local attribute with default value

1. A local attribute which uses an existing type (either a built-in type or a user-defined type), and specifies a default value for the attribute.

```
<attribute  
  name="name"  
  type="type"  
  use="optional"  
  default="default"  
  id="id"  
  form="form"  
>
```

Notes:

- **use**, **id**, and **form** are optional.
- The only legal value for **use** is `optional`. If **use** is not specified then its value defaults to `optional`.
- The type cannot be `ID`, or a type which derives from `ID`.
- Obviously the value for **default** must be of a type compatible with the datatype specified by **type**.

Example 1

schema:

```
<xsd:element name="image">
  <xsd:complexType>
    <xsd:attribute name="src" type="xsd:anyURI" default="http://www.xfront.com"/>
  </xsd:complexType>
</xsd:element>
```

instance:

```
<image/> <!-- Using the default value -->
<image src="http://www.xfront.com/BestPractices.html"/> <!-- Not using the default value -->
```

local attribute with a default, ref's to a global attribute

2. A local attribute which ref's to a global attribute declaration, and specifies a default value for the attribute.

```
<attribute  
  ref="name"  
  use="optional"  
  default="default"  
  id="id"  
>
```

Notes:

- **use**, and **id** are optional.
- The only legal value for **use** is `optional`. If **use** is not specified then its value defaults to `optional`.
- Obviously the value for **default** must be of a type compatible with the datatype specified in the global attribute declaration.

Example 2

schema:

```
<xsd:element name="image">
  <xsd:complexType>
    <xsd:attribute ref="src" default="http://www.xfront.com"/>
  </xsd:complexType>
</xsd:element>
<xsd:attribute name="src" type="xsd:anyURI"/>
```

instance:

```
<image/> <!-- Using the default value -->
<image i:src="http://www.xfront.com/BestPractices.html"/> <!-- Not using the default value -->
```

Note that the attribute must be namespace-qualified since it was declared globally.

global attribute with default value

3. A global attribute which uses an existing type (either a built-in type or a user-defined type), and specifies a default value for the attribute.

```
<attribute  
  name="name"  
  type="type"  
  default="default"  
  id="id"  
>
```

Notes:

- Obviously the value for **default** must be of a type compatible with the datatype specified by **type**.
- The type cannot be ID , or a type which derives from ID.
- **id** is optional.

Example 3

schema:

```
<xsd:element name="image">
  <xsd:complexType>
    <xsd:attribute ref="src"/>
  </xsd:complexType>
</xsd:element>
<xsd:attribute name="src" type="xsd:anyURI" default="http://www.xfront.com"/>
```

instance:

```
<image/> <!-- Using the default value -->
<image i:src="http://www.xfront.com/BestPractices.html"/> <!-- Not using the default value -->
```

Note that the attribute must be namespace-qualified since it was declared globally.

local attribute with fixed value

4. A local attribute which uses an existing type (either a built-in type or a user-defined type), and specifies a fixed value for the attribute.

```
<attribute  
  name="name"  
  type="type"  
  use="optional"  
  fixed="fixed"  
  id="id"  
  form="form"  
>
```

Notes:

- **use**, **id**, and **form** are optional.
- The only legal value for **use** is `optional`. If **use** is not specified then its value defaults to `optional`.
- The type cannot be `ID`, or a type which derives from `ID`.
- Obviously the value for **fixed** must be of a type compatible with the datatype specified by **type**.

Example 4

schema:

```
<xsd:element name="image">  
  <xsd:complexType>  
    <xsd:attribute name="src" type="xsd:anyURI" fixed="http://www.xfront.com"/>  
  </xsd:complexType>  
</xsd:element>
```

instance:

```
<image/> <!-- Using the fixed value -->  
<image src="http://www.xfront.com"/> <!-- Specifying the fixed value -->
```

local attribute with a fixed value, ref's to a global attribute

5. A local attribute which ref's to a global attribute declaration, and specifies a fixed value for the attribute.

```
<attribute  
  ref="name"  
  use="optional"  
  fixed="fixed"  
  id="id"  
>
```

Notes:

- **use**, and **id** are optional.
- The only legal value for **use** is `optional`. If **use** is not specified then its value defaults to `optional`.
- Obviously the value for **fixed** must be of a type compatible with the datatype specified in the global attribute declaration.

Example 5

schema:

```
<xsd:element name="image">
  <xsd:complexType>
    <xsd:attribute ref="src" fixed="http://www.xfront.com"/>
  </xsd:complexType>
</xsd:element>
<xsd:attribute name="src" type="xsd:anyURI"/>
```

instance:

```
<image/> <!-- Using the fixed value -->
<image i:src="http://www.xfront.com"/> <!-- Specifying the fixed value -->
```

Note that the attribute must be namespace-qualified since it was declared globally.

global attribute with fixed value

6. A global attribute which uses an existing type (either a built-in type or a user-defined type), and specifies a fixed value for the attribute.

```
<attribute  
  name="name"  
  type="type"  
  fixed="fixed"  
  id="id"  
>
```

Notes:

- Obviously the value for **fixed** must be of a type compatible with the datatype specified by **type**.
- The type cannot be ID , or a type which derives from ID.
- **id** is optional.

Example 6

schema:

```
<xsd:element name="image">
  <xsd:complexType>
    <xsd:attribute ref="src"/>
  </xsd:complexType>
</xsd:element>
<xsd:attribute name="src" type="xsd:anyURI" fixed="http://www.xfront.com"/>
```

instance:

```
<image/> <!-- Using the fixed value -->
<image i:src="http://www.xfront.com/> <!-- Specifying the fixed value -->
```

Note that the attribute must be namespace-qualified since it was declared globally.

local attribute

7. Attribute which uses an existing type (either a built-in type or a user-defined type). No default or fixed value.

```
<attribute  
  name="name"  
  type="type"  
  use="use"  
  id="id"  
  form="form"  
>
```

Notes:

- The legal values for **use** are: optional, required, and prohibited. If **use** is not specified then its value defaults to optional.
- **use**, **id**, and **form** are optional.
- The two legal values for **form** are qualified and unqualified.

Example 7

schema:

```
<xsd:element name="image">
  <xsd:complexType>
    <xsd:attribute name="src" type="xsd:anyURI"/>
  </xsd:complexType>
</xsd:element>
```

instance:

```
<image/> <!-- src has no value -->
<image src="http://www.xfront.com/BestPractices.html"/> <!-- Assigning src a value -->
```

local attribute, ref's to a global attribute

8. Attribute which ref's to a global attribute declaration. No default or fixed value.

```
<attribute  
  ref="name"  
  use="use"  
  id="id"  
>
```

Notes:

- **use**, and **id** are optional.
- The legal values for **use** are: optional, required, and prohibited. If **use** is not specified then its value defaults to optional.

Example 8

schema:

```
<xsd:element name="image">
  <xsd:complexType>
    <xsd:attribute ref="src"/>
  </xsd:complexType>
</xsd:element>
<xsd:attribute name="src" type="xsd:anyURI"/>
```

instance:

```
<image/> <!-- src has no value -->
<image i:src="http://www.xfront.com/BestPractices.html"/> <!-- Assigning src a value -->
```

Note that the attribute must be namespace-qualified since it was declared globally.

global attribute

9. A global attribute which uses an existing type (either a built-in type or a user-defined type). No default or fixed value.

```
<attribute  
  name="name"  
  type="type"  
  id="id"  
>
```

Notes:

- **id** is optional.

Example 9

schema:

```
<xsd:element name="image">
  <xsd:complexType>
    <xsd:attribute ref="src"/>
  </xsd:complexType>
</xsd:element>
<xsd:attribute name="src" type="xsd:anyURI"/>
```

instance:

```
<image/> <!-- src has no value -->
<image i:src="http://www.xfront.com/BestPractices.html"/> <!-- Assign src a value -->
```

Note that the attribute must be namespace-qualified since it was declared globally.

local attribute with default value

10. A local attribute which defines a simple type inline, and specifies a default value for the attribute.

```
<attribute name="name" use="optional" default="default" id="id" form="form">
  <simpleType>
    ...
  </simpleType>
</attribute>
```

Notes:

- **use**, **id**, and **form** are optional.
- The only legal value for **use** is `optional`. If **use** is not specified then its value defaults to `optional`.
- The `simpleType` cannot be a restriction of ID.
- Obviously the value for **default** must be of a type compatible with the **simpleType**.

Example 10

schema:

```
<xsd:element name="image">
  <xsd:complexType>
    <xsd:attribute name="src" default="http://www.xfront.com">
      <xsd:simpleType>
        <xsd:restriction base="xsd:anyURI">
          <xsd:enumeration value="http://www.xfront.com"/>
          <xsd:enumeration value="http://www.xfront.com/BestPractices.html"/>
          <xsd:enumeration value="http://www.xfront.com/isbn.html"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
  </xsd:complexType>
</xsd:element>
```

instance:

```
<image/> <!-- Using the default value -->
<image src="http://www.xfront.com/BestPractices.html"/> <!-- Not using the default value -->
```

local attribute with a default, ref's to a global attribute

11. A local attribute which ref's to a global attribute declaration, and specifies a default value for the attribute.

```
<attribute ref="name" use="optional" default="default" id="id"/>
```

Notes:

- **use**, and **id** are optional.
- The only legal value for **use** is `optional`. If **use** is not specified then its value defaults to `optional`.
- Obviously the value for **default** must be of a type compatible with the datatype specified in the global attribute declaration.

Example 11

schema:

```
<xsd:element name="image">
  <xsd:complexType>
    <xsd:attribute ref="src" default="http://www.xfront.com"/>
  </xsd:complexType>
</xsd:element>

<xsd:attribute name="src">
  <xsd:simpleType>
    <xsd:restriction base="xsd:anyURI">
      <xsd:enumeration value="http://www.xfront.com"/>
      <xsd:enumeration value="http://www.xfront.com/BestPractices.html"/>
      <xsd:enumeration value="http://www.xfront.com/isbn.html"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:attribute>
```

instance:

```
<image/> <!-- Using the default value -->
<image i:src="http://www.xfront.com/BestPractices.html"/> <!-- Not using the default value -->
```

Note that the attribute must be namespace-qualified since it was declared globally.

global attribute with default value

12. A global attribute which defines a simple type inline, and specifies a default value for the attribute.

```
<attribute name="name" default="default" id="id">  
  <simpleType>  
    ...  
  </simpleType>  
</attribute>
```

Notes:

- Obviously the value for **default** must be of a type compatible with the **simpleType**.
- The **simpleType** cannot be a restriction of **ID**.
- **id** is optional.

Example 12

schema:

```
<xsd:element name="image">
  <xsd:complexType>
    <xsd:attribute ref="src"/>
  </xsd:complexType>
</xsd:element>

<xsd:attribute name="src" default="http://www.xfront.com">
  <xsd:simpleType>
    <xsd:restriction base="xsd:anyURI">
      <xsd:enumeration value="http://www.xfront.com"/>
      <xsd:enumeration value="http://www.xfront.com/BestPractices.html"/>
      <xsd:enumeration value="http://www.xfront.com/isbn.html"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:attribute>
```

instance:

```
<image/> <!-- Using the default value -->
<image i:src="http://www.xfront.com/BestPractices.html"/> <!-- Not using the default value -->
```

Note that the attribute must be namespace-qualified since it was declared globally.

local attribute with fixed value

13. A local attribute which defines a simple type inline, and specifies a fixed value for the attribute.

```
<attribute name="name" use="optional" fixed="fixed" id="id" form="form">  
  <simpleType>  
    ...  
  </simpleType>  
</attribute>
```

Notes:

- **use**, **id**, and **form** are optional.
- The only legal value for **use** is `optional`. If **use** is not specified then its value defaults to `optional`.
- The `simpleType` cannot be a restriction of ID.
- Obviously the value for **fixed** must be of a type compatible with the **simpleType**.

Example 13

schema:

```
<xsd:element name="image">
  <xsd:complexType>
    <xsd:attribute name="src" fixed="http://www.xfront.com">
      <xsd:simpleType>
        <xsd:restriction base="xsd:anyURI">
          <xsd:enumeration value="http://www.xfront.com"/>
          <xsd:enumeration value="http://www.xfront.com/BestPractices.html"/>
          <xsd:enumeration value="http://www.xfront.com/isbn.html"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
  </xsd:complexType>
</xsd:element>
```

instance:

```
<image/> <!-- Using the fixed value -->
<image src="http://www.xfront.com"/> <!-- Specifying the fixed value -->
```

local attribute with a fixed value, ref's to a global attribute

14. A local attribute which ref's to a global attribute declaration, and specifies a fixed value for the attribute.

```
<attribute ref="name" use="optional" fixed="fixed" id="id"/>
```

Notes:

- **use**, and **id** are optional.
- The only legal value for **use** is `optional`. If **use** is not specified then its value defaults to `optional`.
- Obviously the value for **fixed** must be of a type compatible with the datatype specified in the global attribute declaration.

Example 14

schema:

```
<xsd:element name="image">
  <xsd:complexType>
    <xsd:attribute ref="src" fixed="http://www.xfront.com"/>
  </xsd:complexType>
</xsd:element>

<xsd:attribute name="src">
  <xsd:simpleType>
    <xsd:restriction base="xsd:anyURI">
      <xsd:enumeration value="http://www.xfront.com"/>
      <xsd:enumeration value="http://www.xfront.com/BestPractices.html"/>
      <xsd:enumeration value="http://www.xfront.com/isbn.html"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:attribute>
```

instance:

```
<image/>                                <!-- Using the fixed value -->
<image i:src="http://www.xfront.com"/> <!-- Specifying the fixed value -->
```

Note that the attribute must be namespace-qualified since it was declared globally.

global attribute with fixed value

15. A global attribute which creates a user-defined type inline, and specifies a fixed value for the attribute.

```
<attribute name="name" fixed="fixed" id="id">  
  <simpleType>  
    ...  
  </simpleType>  
</attribute>
```

Notes:

- Obviously the value for **fixed** must be of a type compatible with the **simpleType**.
- The **simpleType** cannot be a restriction of **ID**.
- **id** is optional.

Example 15

schema:

```
<xsd:element name="image">
  <xsd:complexType>
    <xsd:attribute ref="src"/>
  </xsd:complexType>
</xsd:element>

<xsd:attribute name="src" fixed="http://www.xfront.com">
  <xsd:simpleType>
    <xsd:restriction base="xsd:anyURI">
      <xsd:enumeration value="http://www.xfront.com"/>
      <xsd:enumeration value="http://www.xfront.com/BestPractices.html"/>
      <xsd:enumeration value="http://www.xfront.com/isbn.html"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:attribute>
```

instance:

```
<image/>                                <!-- Using the fixed value -->
<image i:src="http://www.xfront.com"/> <!-- Specifying the fixed value -->
```

Note that the attribute must be namespace-qualified since it was declared globally.

local attribute

16. A local attribute which creates a user-defined type inline.
No default or fixed value.

```
<attribute name="name" use="use" id="id" form="form">  
  <simpleType>  
    ...  
  </simpleType>  
</attribute>
```

- Notes:
- The legal values for **use** are: `optional`, `required`, and `prohibited`. If **use** is not specified then its value defaults to `optional`.
 - **use**, **id**, and **form** are optional.

Example 16

schema:

```
<xsd:element name="image">
  <xsd:complexType>
    <xsd:attribute name="src">
      <xsd:simpleType>
        <xsd:restriction base="xsd:anyURI">
          <xsd:enumeration value="http://www.xfront.com"/>
          <xsd:enumeration value="http://www.xfront.com/BestPractices.html"/>
          <xsd:enumeration value="http://www.xfront.com/isbn.html"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
  </xsd:complexType>
</xsd:element>
```

instance:

```
<image/> <!-- src has no value -->
<image src="http://www.xfront.com"/> <!-- Assigning src a value -->
```

local attribute ref's to a global attribute

17. A local attribute which ref's to a global attribute declaration.
No default or fixed value.

```
<attribute ref="name" use="use" id="id"/>
```

Notes:

- **use**, and **id** are optional.
- The legal values for **use** are: optional, required, and prohibited. If **use** is not specified then its value defaults to optional.

Example 17

schema:

```
<xsd:element name="image">
  <xsd:complexType>
    <xsd:attribute ref="src"/>
  </xsd:complexType>
</xsd:element>

<xsd:attribute name="src">
  <xsd:simpleType>
    <xsd:restriction base="xsd:anyURI">
      <xsd:enumeration value="http://www.xfront.com"/>
      <xsd:enumeration value="http://www.xfront.com/BestPractices.html"/>
      <xsd:enumeration value="http://www.xfront.com/isbn.html"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:attribute>
```

instance:

```
<image/> <!-- src has no value -->
<image i:src="http://www.xfront.com"/> <!-- Assigning src a value -->
```

Note that the attribute must be namespace-qualified since it was declared globally.

global attribute

18. A global attribute which creates a user-defined type inline
No default or fixed value.

```
<attribute name="name" id="id">  
  <simpleType>  
    ...  
  </simpleType>  
</attribute>
```

Notes:

- **id** is optional.

Example 18

schema:

```
<xsd:element name="image">
  <xsd:complexType>
    <xsd:attribute ref="src"/>
  </xsd:complexType>
</xsd:element>

<xsd:attribute name="src">
  <xsd:simpleType>
    <xsd:restriction base="xsd:anyURI">
      <xsd:enumeration value="http://www.xfront.com"/>
      <xsd:enumeration value="http://www.xfront.com/BestPractices.html"/>
      <xsd:enumeration value="http://www.xfront.com/isbn.html"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:attribute>
```

instance:

```
<image/> <!-- src has no value -->
<image i:src="http://www.xfront.com"/> <!-- Assigning src a value -->
```

↑
Note that the attribute is
namespace-qualified since it is global

annotating the attribute

19. For each of the previous forms, we can inline an annotation element

```
<attribute name="name" type="type" ... >  
  <annotation>  
    ...  
  </annotation>  
</attribute>
```

```
<attribute ref="name" ... >  
  <annotation>  
    ...  
  </annotation>  
</attribute>
```

```
<attribute name="name" ... >  
  <annotation>  
    ...  
  </annotation>  
  <simpleType>  
    ...  
  </simpleType>  
</attribute>
```

non-native attributes

20. For each of the previous forms, we can add non-native attributes.

```
<attribute name="name" type="type" ... other:x="x" other:y="y">  
  <annotation>  
    ...  
  </annotation>  
</attribute>
```

```
<attribute ref="name" ... other:x="x" other:y="y">  
  <annotation>  
    ...  
  </annotation>  
</attribute>
```

```
<attribute name="name" ... other:x="x" other:y="y">  
  <annotation>  
    ...  
  </annotation>  
  <simpleType>  
    ...  
  </simpleType>  
</attribute>
```

Note: the non-native
attributes must come
from a namespace other
than
<http://.../XMLSchema>

use="prohibited"

Question: When would **use="prohibited"** be used?

Answer: This value is useful when you create a master type which lists all possible attributes, and then subtypes can delete the attributes that are not needed.

Here's an example:

```
<xsd:complexType name="shape">
  <xsd:attribute name="length" type="xsd:nonNegativeInteger"/>
  <xsd:attribute name="height" type="xsd:nonNegativeInteger"/>
  <xsd:attribute name="width" type="xsd:nonNegativeInteger"/>
  <xsd:attribute name="radius" type="xsd:nonNegativeInteger"/>
  <xsd:attribute name="diameter" type="xsd:nonNegativeInteger"/>
</xsd:complexType>
```

```
<xsd:complexType name="box">
  <xsd:complexContent>
    <xsd:restriction base="shape">
      <xsd:attribute name="length" type="xsd:nonNegativeInteger"/>
      <xsd:attribute name="height" type="xsd:nonNegativeInteger"/>
      <xsd:attribute name="width" type="xsd:nonNegativeInteger"/>
      <xsd:attribute name="radius" type="xsd:nonNegativeInteger" use="prohibited"/>
      <xsd:attribute name="diameter" type="xsd:nonNegativeInteger" use="prohibited"/>
    </xsd:restriction>
  </xsd:complexContent>
</xsd:complexType>
```

No datatype specified

You can declare an attribute without specifying a datatype. For example:

```
<attribute name="shape"/>
```

This attribute is unconstrained with respect to the type of value it can have.

Creating unconstrained attributes is a very useful technique:

Consider creating a complexType containing unconstrained attributes, and then create other complexTypes which derive by restriction and which constrain the shape attribute to a specific datatype. That's nice!

Final notes about attributes

1. It is illegal to declare an attribute whose name="xmlns".

This attribute is built-in to schema validators. So, you can use it in instance documents without declaring it in your schema.

2. nil, schemaLocation, noNamespaceSchemaLocation, and type are attributes in the `http://www.w3.org/2001/XMLSchema-instance` namespace. You can use these attributes in your instance documents without declaring them in your schema.

3. The name that you give to an attribute, cannot have a colon in it, e.g, `<attribute name="air:velocity" .../>` is not legal.

4. You now know everything there is to know about attributes! Give yourself a pat on the back.

Items used to declare elements

Here are items that you use in declaring elements:

name: use this to declare what will be the name of the element in instance documents.

ref: use this to refer-to an element declaration.

type: use this to declare the datatype of the element's value in instance documents.

id: use this to associate a unique identifier to the element component. This id has no representation in the instance document. It is purely internal to the schema. The type of this value must be `xsd:ID`.

form: use this to override **elementFormDefault** (which is specified on the schema element). There are two possible values - `qualified` or `unqualified`. If **form** is assigned the value `qualified` then the element must be namespace-qualified in instance documents.

abstract: use this to indicate that the element is only a placeholder for other elements which are in its substitutionGroup. The default value for **abstract** is `false`.

Cont. -->

Items used to declare elements

Continued ...

- block:** use this to prohibit this element from being replaced by an element in its substitutionGroup, or prohibit this element's type from being replaced by a derived type.
- default:** use this to give the element a default value.
- fixed:** use this to give the element a fixed (constant) value.
- minOccurs:** use this to specify the minimum number of times that the element may occur in an instance document. The default value for **minOccurs** is 1.
- maxOccurs:** use this to specify the maximum number of times that the element may occur in an instance document. The default value for **maxOccurs** is 1. Note: the value for **minOccurs** must be less than or equal to **maxOccurs**.
- nillable:** use this to assert that if there's no value available for the element then in the instance document we indicate it with the xsi:null attribute. The default value for **nillable** is false.
- substitutionGroup:** use this to indicate that this element can substitute for the element identified by the substitutionGroup.

global versus local elements

Elements may be declared locally or globally:

1. Ways to create local element declarations:
 - 1.1 Inline an element declaration within a complexType.
 - 1.2 Inline an element declaration within a group
2. Global element declarations are immediate children of <schema>

```
<xsd:complexType name="Publication">
  <xsd:sequence>
    <xsd:element name="Title" type="xsd:string"/> 1.1
    <xsd:element ref="Author"/>
    <xsd:group ref="bookElements"/>
  </xsd:sequence>
</xsd:complexType>
<xsd:element name="Author" type="xsd:string"/> 2.
<xsd:group name="bookElements">
  <xsd:element name="Date" type="xsd:date"/> 1.2
  <xsd:element name="ISBN" type="xsd:string1"/>
</xsd:group>
```

elementFormDefault versus form

< **schema**> has an optional attribute, **elementFormDefault**.

Example: <xsd:**schema** ...**elementFormDefault**="qualified" ...>

The legal values for **elementFormDefault** are `qualified` and `unqualified`.

The default value of **elementFormDefault** is `unqualified`.

If **elementFormDefault**="unqualified" then local elements must not be namespace-qualified in instance documents.

If **elementFormDefault**="qualified" then all elements must be namespace-qualified in instance documents.

An element declaration may override **elementFormDefault** with **form**:

<xsd:**element** ... **form**="qualified"/> This element must be namespace-qualified in instance documents.

When must elements be namespace-qualified?

`<a:aircraft>Boeing 747</a:aircraft>`



An element must be namespace-qualified in instance documents if:

- when the element (e.g., aircraft) was declared you specified **form**="qualified", or
- the element was declared globally, or
- there was no value specified for **form** when the element was declared, but the schema has: `<schema ... elementFormDefault="qualified"...>`

17 ways to declare elements

- On the following slides I have identified 17 different ways to declare elements.
- With each way to declare an element I have provided an example.
 - The folder `example38` contains all the examples.

local element with default value

1. A local element which uses an existing type (either a built-in type or a user-defined type), and specifies a default value for the element.

```
<element
  name="name"
  type="type"
  default="default"
  minOccurs="min"
  maxOccurs="max"
  block="block"
  nillable="boolean"
  id="id"
  form="form"
/>
```

Notes:

- **type** must specify a simple datatype.
- **minOccurs**, **maxOccurs**, **block**, **nillable**, **id**, and **form** are optional.
- Obviously the value for **default** must be of a type compatible with the datatype specified by **type**.

Example 1

schema:

```
<xsd:element name="ElementExamples">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="aircraft" default="Boeing 747" maxOccurs="2" type="xsd:string"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

instance:

```
<aircraft/> <!-- Using the default value -->
<aircraft>F-16</aircraft> <!-- Not using the default value -->
```

global element with default value

2. A global element which uses an existing type (either a built-in type or a user-defined type), and specifies a default value for the element.

```
<element  
  name="name"  
  type="type"  
  default="default"  
  abstract="boolean"  
  substitutionGroup="name"  
  block="block"  
  final="final"  
  nillable="boolean"  
  id="id"  
>
```

Notes:

- **type** must specify a simple datatype.
- **abstract**, **substitutionGroup**, **block**, **final**, **nillable**, and **id** are optional.
- if **abstract**="true" then **block** cannot have the value `substitutionGroup`.
- Obviously the value for **default** must be of a type compatible with the datatype specified by **type**.

Example 2

schema:

```
<xsd:element name="aircraft" default="Boeing 747" type="xsd:string"/>
<xsd:element name="ElementExamples">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="aircraft" maxOccurs="2"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

instance:

```
<aircraft/> <!-- Using the default value -->
<aircraft>F-16</aircraft> <!-- Not using the default value -->
```

local element with fixed value

3. A local element which uses an existing type (either a built-in type or a user-defined type), and specifies a fixed value for the element.

```
<element  
  name="name"  
  type="type"  
  fixed="fixed"  
  minOccurs="min"  
  maxOccurs="max"  
  block="block"  
  id="id"  
  form="form"
```

```
/>
```

Notes:

- **type** must specify a simple datatype.
- **minOccurs**, **maxOccurs**, **block**, **id**, and **form** are optional.
- Obviously the value for **fixed** must be of a type compatible with the datatype specified by **type**.

Example 3

schema:

```
<xsd:element name="ElementExamples">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="aircraft" fixed="Boeing 747" maxOccurs="2" type="xsd:string"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

instance:

```
<aircraft/> <!-- Using the fixed value -->
<aircraft>Boeing 747</aircraft> <!-- Specifying the fixed value -->
```

global element with fixed value

4. A global element which uses an existing type (either a built-in type or a user-defined type), and specifies a fixed value for the element.

```
<element  
  name="name"  
  type="type"  
  fixed="fixed"  
  abstract="boolean"  
  substitutionGroup="name"  
  block="block"  
  final="final"  
  id="id"  
>
```

Notes:

- **type** must specify a simple datatype.
- **abstract**, **substitutionGroup**, **block**, **final**, and **id** are optional.
- if **abstract**="true" then **block** cannot have the value `substitutionGroup`.
- Obviously the value for **fixed** must be of a type compatible with the datatype specified by **type**.

Example 4

schema:

```
<xsd:element name="aircraft" fixed="Boeing 747" type="xsd:string"/>
<xsd:element name="ElementExamples">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="aircraft" maxOccurs="2"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

instance:

```
<aircraft/> <!-- Using the fixed value -->
<aircraft>Boeing 747</aircraft> <!-- Specifying the fixed value -->
```

local element

(no default/fixed value)

5. A local element which uses an existing type (either a simple type or a complex type). No default or fixed value.

```
<element  
  name="name"  
  type="type"  
  minOccurs="min"  
  maxOccurs="max"  
  block="block"  
  nillable="boolean"  
  id="id"  
  form="form"  
>
```

Notes:

- **type** can specify either a simple type or the name of a complexType.
- **minOccurs**, **maxOccurs**, **block**, **nillable**, **id**, and **form** are optional.

Example 5

schema:

```
<xsd:element name="ElementExamples">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="aircraft" maxOccurs="2" type="xsd:string"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

instance:

```
<aircraft>F-16</aircraft> <!-- Assigning aircraft a value -->
```

local element, ref's to a global element

6. A local element which ref's to a global element declaration. No default or fixed value.

```
<element  
  ref="name"  
  minOccurs="min"  
  maxOccurs="max"  
  id="id"  
>
```

Notes:

- **minOccurs**, **maxOccurs**, and **id** are optional.

Example 6

schema:

```
<xsd:element name="aircraft" type="xsd:string"/>
<xsd:element name="ElementExamples">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="aircraft" maxOccurs="2"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

instance:

```
<aircraft>F-16</aircraft> <!-- Assigning aircraft a value -->
```

global element (no default/fixed value)

7. A global element which uses an existing type (either a simple type or a complex type). No default or fixed value.

```
<element  
  name="name"  
  type="type"  
  abstract="boolean"  
  substitutionGroup="name"  
  block="block"  
  final="final"  
  nillable="boolean"  
  id="id"
```

```
/>
```

Notes:

- **type** can specify the name of either a simple type or a complex type.
- **abstract**, **substitutionGroup**, **block**, **final**, **nillable**, and **id** are optional.
- if **abstract**="true" then **block** cannot have the value `substitutionGroup`.

Example 7

schema:

```
<xsd:element name="aircraft" type="xsd:string"/>
<xsd:element name="ElementExamples">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="aircraft" maxOccurs="2"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

instance:

```
<aircraft>F-16</aircraft> <!-- Assigning aircraft a value -->
```

local element with default value

8. A local element which defines a simple type inline, and specifies a default value for the element.

```
<element name="name" default="default" minOccurs="min" maxOccurs="max" block="block" nillable="nillable" id="id" form="form">  
  <simpleType>  
    ...  
  </simpleType>  
</element>
```

Notes:

- **minOccurs**, **maxOccurs**, **block**, **nillable**, **id**, and **form** are optional.
- Obviously the value for **default** must be of a type compatible with the datatype specified by the `simpleType`.

Example 8

schema:

```
<xsd:element name="ElementExamples">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="aircraft" default="Boeing 747" maxOccurs="2">
        <xsd:simpleType>
          <xsd:restriction base="xsd:string">
            <xsd:enumeration value="Boeing 747"/>
            <xsd:enumeration value="F-16"/>
            <xsd:enumeration value="A-10"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

instance:

```
<aircraft/> <!-- Using the default value -->
<aircraft>F-16</aircraft> <!-- Not using the default value -->
```

global element with default value

9. A global element which defines a simple type inline, and specifies a default value for the element.

```
<element name="name" default="default" abstract="boolean" substitutionGroup="name" block="block" final="final" nillable="nillable" id="id">  
  <simpleType>  
    ...  
  </simpleType>  
</element>
```

Notes:

- **abstract**, **substitutionGroup**, **block**, **final**, **nillable**, and **id** are optional.
- if **abstract**="true" then **block** cannot have the value `substitutionGroup`.
- Obviously the value for **default** must be of a type compatible with the datatype specified by the `simpleType`.

Example 9

schema:

```
<xsd:element name="aircraft" default="Boeing 747">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="Boeing 747"/>
      <xsd:enumeration value="F-16"/>
      <xsd:enumeration value="A-10"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>

<xsd:element name="ElementExamples">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="aircraft" maxOccurs="2"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

instance:

```
<aircraft/> <!-- Using the default value -->
<aircraft>F-16</aircraft> <!-- Not using the default value -->
```

local element with fixed value

10. A local element which defines a simple type inline, and specifies a fixed value for the element.

```
<element name="name" fixed="fixed" minOccurs="min" maxOccurs="max" block="block" id="id" form="form">  
  <simpleType>  
    ...  
  </simpleType>  
</element>
```

Notes:

- **abstract**, **substitutionGroup**, **block**, **final**, and **id** are optional.
- if **abstract**="true" then **block** cannot have the value `substitutionGroup`.
- Obviously the value for **fixed** must be of a type compatible with the datatype specified by the `simpleType`.

Example 10

schema:

```
<xsd:element name="ElementExamples">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="aircraft" fixed="Boeing 747" maxOccurs="2">
        <xsd:simpleType>
          <xsd:restriction base="xsd:string">
            <xsd:enumeration value="Boeing 747"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

instance:

```
<aircraft/> <!-- Using the fixed value -->
<aircraft>Boeing 747</aircraft> <!-- Specifying the fixed value -->
```

global element with fixed value

11. A global element which defines a simple type inline, and specifies a fixed value for the element.

```
<element name="name" fixed="fixed" abstract="boolean" substitutionGroup="name" block="block" final="final" id="id">  
  <simpleType>  
    ...  
  </simpleType>  
</element>
```

Notes:

- **abstract**, **substitutionGroup**, **block**, **final**, and **id** are optional.
- if **abstract**="true" then **block** cannot have the value `substitutionGroup`.
- Obviously the value for **fixed** must be of a type compatible with the datatype specified by the `simpleType`.

Example 11

schema:

```
<xsd:element name="aircraft" fixed="Boeing 747">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="Boeing 747"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>

<xsd:element name="ElementExamples">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="aircraft" maxOccurs="2"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

instance:

```
<aircraft/> <!-- Using the fixed value -->
<aircraft>Boeing 747</aircraft> <!-- Specifying the fixed value -->
```

local element with simpleType

12. A local element which defines a simple type inline.
No default or fixed value.

```
<element name="name" minOccurs="min" maxOccurs="max" block="block" nillable="nillable" id="id" form="form">  
  <simpleType>  
    ...  
  </simpleType>  
</element>
```

Notes:

- **minOccurs**, **maxOccurs**, **block**, **nillable**, **id**, and **form** are optional.

Example 12

schema:

```
<xsd:element name="ElementExamples">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="aircraft" maxOccurs="2">
        <xsd:simpleType>
          <xsd:restriction base="xsd:string">
            <xsd:enumeration value="Boeing 747"/>
            <xsd:enumeration value="F-16"/>
            <xsd:enumeration value="A-10"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

instance:

```
<aircraft>F-16</aircraft> <!-- Assigning aircraft a value -->
```

global element with simpleType

13. A global element which defines a simple type inline.
No default or fixed value

```
<element name="name" abstract="boolean" substitutionGroup="name" block="block" final="final" nillable="nillable" id="id">  
  <simpleType>  
    ...  
  </simpleType>  
</element>
```

Notes:

- **abstract**, **substitutionGroup**, **block**, **final**, **nillable**, and **id** are optional.
- if **abstract**="true" then **block** cannot have the value `substitutionGroup`.

Example 13

schema:

```
<xsd:element name="aircraft">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="Boeing 747"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>

<xsd:element name="ElementExamples">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="aircraft" maxOccurs="2"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

instance:

```
<aircraft>F-16</aircraft> <!-- Assigning aircraft a value -->
```

local element with complexType

14. A local element which defines a complexType inline.

```
<element name="name" minOccurs="min" maxOccurs="max" block="block" nillable="nillable" id="id" form="form">  
  <complexType>  
    ...  
  </complexType>  
</element>
```

Notes:

- **minOccurs**, **maxOccurs**, **block**, **nillable**, **id**, and **form** are optional.

Example 14

schema:

```
<xsd:element name="ElementExamples">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="aircraft" maxOccurs="2">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="type" type="xsd:string"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

instance:

```
<aircraft>
  <type>F-16</type>
</aircraft>
```

global element with complexType

15. A global element which defines a complexType inline.

```
<element name="name" abstract="boolean" substitutionGroup="name" block="block" final="final" nillable="nillable" id="id">  
  <complexType>  
    ...  
  </complexType>  
</element>
```

Notes:

- **abstract**, **substitutionGroup**, **block**, **final**, **nillable**, and **id** are optional.
- if **abstract**="true" then **block** cannot have the value `substitutionGroup`.

Example 15

schema:

```
<xsd:element name="aircraft">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="type" type="xsd:string"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>

<xsd:element name="ElementExamples">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="aircraft" maxOccurs="2"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

instance:

```
<aircraft>
  <type>F-16</type>
</aircraft>
```

annotating the element

16. For each of the previous forms, we can inline an annotation element

```
<element name="name" type="type" ... >  
  <annotation>  
    ...  
  </annotation>  
</element>
```

```
<element name="name" ... >  
  <annotation>  
    ...  
  </annotation>  
  <simpleType>  
    ...  
  </simpleType>  
</element>
```

```
<element ref="name" ... >  
  <annotation>  
    ...  
  </annotation>  
</element>
```

```
<element name="name" ... >  
  <annotation>  
    ...  
  </annotation>  
  <complexType>  
    ...  
  </complexType>  
</element>
```

non-native attributes

17. For each of the previous forms, we can add non-native attributes.

```
<element name="name" type="type" ... other:x="x" other:y="y">
  <annotation>
    ...
  </annotation>
</element>
```

```
<element ref="name" ... other:x="x" other:y="y">
  <annotation>
    ...
  </annotation>
</element>
```

Note: the non-native attributes must come from a namespace other than `http://.../XMLSchema`

```
<element name="name" ... other:x="x" other:y="y">
  <annotation>
    ...
  </annotation>
  <simpleType>
    ...
  </simpleType>
</element>
```

```
<element name="name" ... other:x="x" other:y="y">
  <annotation>
    ...
  </annotation>
  <complexType>
    ...
  </complexType>
</element>
```

complexType

Coming soon ...

simpleType

Coming soon ...

group

Coming soon ...

attributeGroup

Coming soon ...

annotation

Coming soon ...