XML Schema Definition

(XSD)

Introduction to XML Schema

- XML-based alternative to DTD.
- describes the structure of an XML document.
- also referred to as XML Schema Definition (XSD).

XML Schema 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>
```

What is an XML Schema?

The purpose of an XML Schema is to define

- the legal **building blocks** of an XML document, just like a DTD
- **elements** that can appear in a document
- **attributes** that can appear in a document
- which elements are **child elements**
- the **order** of child elements
- the **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

What is an XML Schema?

XML Schema is

- the **successors** of DTD
- **extensible** to future additions
- **richer and more powerful** than DTDs
- **written in XML**

XML Schemas support

- **data types**
- **namespaces**

XML Schema Support Data Types

It is easier to

- **describe** allowable document content
- **validate** the correctness of data
- work with data from a **database**
- define **data facets** (restrictions on data)
- define data **patterns** (data formats)
- **convert data** between different data types

XML Schema use XML Syntax

- 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

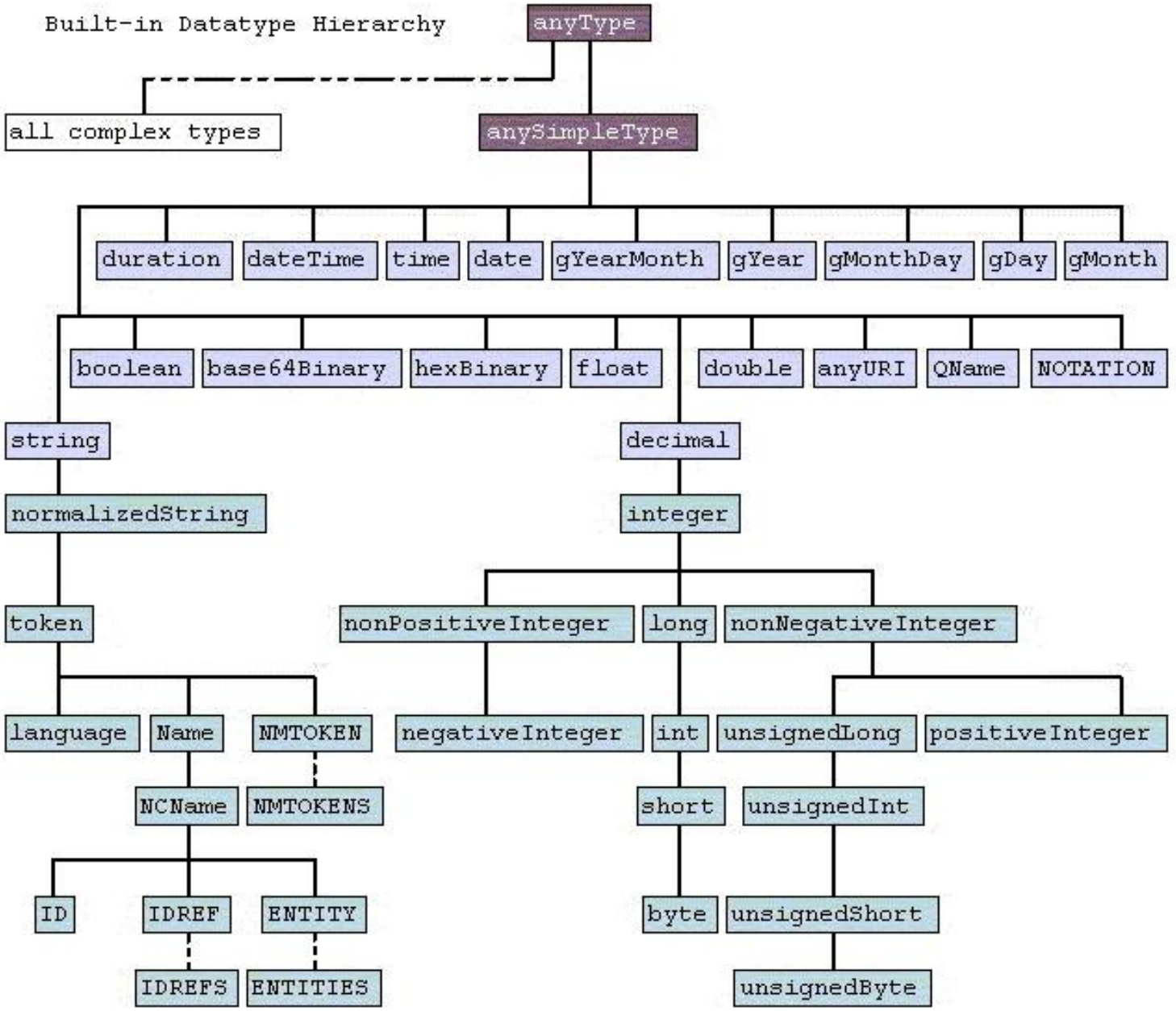
XML Schema Secure Data Communication

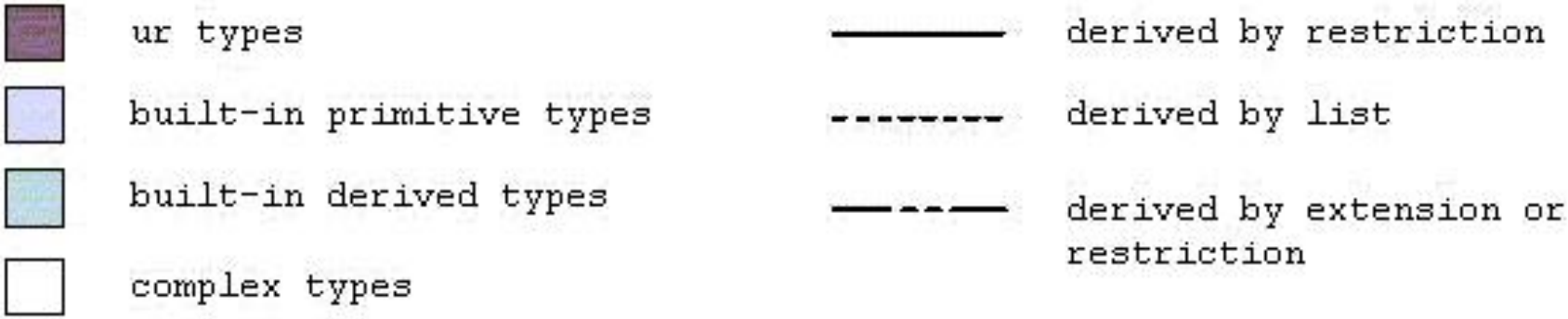
- When sending data from a sender to a receiver, it is essential that both parts have the same "expectations" about the content.
- A date like: "03-11-2004" will, in some countries, be interpreted as 3.November and in other countries as 11.March.
- However, an XML element with a data type like this:
- **<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**".

XML Schemas are Extensible

- **Reuse** your Schema in other Schema
- Create your own data types derived from the standard types
- Reference multiple schema in the same document

3 Built-in datatypes




ur types
built-in primitive types
built-in derived types
complex types
derived by restriction
derived by list
derived by extension or restriction

A Simple XML Document

"note.xml":

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

A Reference to a DTD (note_dtd.xml)

```
<?xml version="1.0"?>
<!DOCTYPE note SYSTEM "http://www.w3schools.com/dtd/note.dtd">
<note>
  <to>Tove</to>
  <from>Jane</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

A DTD File (note.dtd)

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

note.xml

```
<?xml version="1.0"?>
<note
xmlns="http://www.w3schools.com"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.w3schools.com note.xsd">

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

xmlns="http://www.w3schools.com"

- specifies the default namespace declaration. This declaration tells the schema-validator that all the elements used in this XML document are declared in the "http://www.w3schools.com" namespace.

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

- XML Schema Instance namespace

xsi:schemaLocation="http://www.w3schools.com note.xsd"

- you can use the schemaLocation attribute. This attribute has two values, separated by a space. The first value is the namespace to use. The second value is the location of the XML schema to use for that namespace:

note.xsd

```
<?xml version="1.0"?>
<xs:schema
xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.w3schools.com"
xmlns="http://www.w3schools.com"
elementFormDefault="qualified">
<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>
```

XSD - The <schema> Element

- The <schema> element is the root element of every XML Schema.

xmlns:xs="http://www.w3.org/2001/XMLSchema"

- indicates that the elements and data types used in the schema come from the "http://www.w3.org/2001/XMLSchema" namespace.
- It also specifies that the elements and data types that come from the "http://www.w3.org/2001/XMLSchema" namespace should be prefixed with **xs:**

targetNamespace="http://www.w3schools.com"

•indicates that the elements defined by this schema (note, to, from, heading, body.) come from the "http://www.w3schools.com" namespace.

xmlns="http://www.w3schools.com"

•indicates that the default namespace is "http://www.w3schools.com".

elementFormDefault="qualified"

•indicates that any elements used by the XML instance document which were declared in this schema must be namespace qualified.

Consider the following ComplexType `AuthorType` used by `author` element

```
<xsd:complexType name="AuthorType">
  <!-- compositor goes here -->
  <xsd:sequence>
     <xsd:element name="name" type="xsd:string"/>
     <xsd:element name="phone" type="tns:Phone"/>
  </xsd:sequence>
  <xsd:attribute name="id" type="tns:AuthorId"/>
</xsd:complexType>
<xsd:element name="author" type="tns:AuthorType"/>
```

If `elementFormDefault="unqualified"`

then following XML Instance is valid

```
<x:author xmlns:x="http://example.org/publishing">
   <name>Aaron Skonnard</name>
   <phone>(801)390-4552</phone>
</x:author>
```

the authors's name attribute is allowed without specifying the namespace(unqualified). Any elements which are a part of `<xsd:complexType>` are considered as local to complexType.

if `elementFormDefault="qualified"`

then the instance should have the local elements qualified

```
<x:author xmlns:x="http://example.org/publishing">
   <x:name>Aaron Skonnard</name>
   <x:phone>(801)390-4552</phone>
</x:author>
```

If you don‘t add elementFormDefault="qualified“ to xsd:schema, then the default unqualified value means that locally declared elements are in **no namespace**.

XSD Simple Elements

•**A simple element is an XML element that contains only text. It cannot contain any other elements or attributes.**

•The text can be of many different **types**. It can be one of the types included in the XML Schema definition (**boolean, string, date**, etc.), or it can be a **custom type** that you can define yourself.

Defining a Simple Element

<xs:element name="xxx" type="yyy"/>

XML Schema has a lot of built-in data types.

•xs:string, xs:decimal, xs:integer, xs:boolean, xs:date, xs:time

Example

Here are some XML elements:

```
<lastname>    Refsnes        </lastname>
<age>         36             </age>
<dateborn>    1970-03-27     </dateborn>
```

And here are the corresponding simple element definitions:

```
<xs:element   name="lastname"     type="xs:string"/>
<xs:element   name="age"          type="xs:integer"/>
<xs:element   name="dateborn"     type="xs:date"/>
```

Default and Fixed Values for Simple Elements

- A **default value** is automatically assigned to the element when no other value is specified.

```
<xs:element name="color" type="xs:string" default="red"/>
```

- A **fixed value** is also automatically assigned to the element, and you cannot specify another value.

```
<xs:element name="color" type="xs:string" fixed="red"/>
```

XSD Attributes

What is an Attribute?

Simple elements cannot have attributes. If an element has attributes, it is considered to be of a complex type. But the attribute itself is always declared as a simple type.

How to Define an Attribute?

- <xs:attribute name="xxx" type="yyy"/>

Examples

Here is an XML element with an attribute:

<lastname lang="EN">Smith</lastname>

And here is the corresponding attribute definition:

<xs:attribute name="lang" type="xs:string"/>

Default and Fixed Values for Attributes

A default value is automatically assigned to the attribute when no other value is specified.

```
<xs:attribute name="lang" type="xs:string" default="EN"/>
```

A fixed value is also automatically assigned to the attribute, and you cannot specify another value.

```
<xs:attribute name="lang" type="xs:string" fixed="EN"/>
```

Optional and Required Attributes

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

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

Restrictions on Content

- When an XML element or attribute has a data type defined, it puts restrictions on the element's or attribute's content

- If an XML element is of type "xs:date" and contains a string like "Hello World", the element will not validate

- With XML Schema, you can also add your own restrictions to your XML elements and attributes. These restrictions are called **facets**

Restrictions on a Set of Values

•To limit the content of an XML element to a set of acceptable values, we would use the enumeration constraint.

```
<xs:element name="car">
    <xs:simpleType>
        <xs:restriction    base="xs:string">
            <xs:enumeration   value="Audi"/>
            <xs:enumeration   value="Golf"/>
            <xs:enumeration   value="BMW"/>
        </xs:restriction>
    </xs:simpleType>
</xs:element>
```

Restrictions on a Set of Values

The whiteSpace constraint is set to "preserve", which means that the XML processor WILL NOT remove any white space characters:

```
<xs:element name="address">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:whiteSpace value="preserve"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

Restrictions on a Set of Values

The whiteSpace constraint is set to "replace", which means that the XML processor WILL REPLACE all white space characters (line feeds, tabs, spaces, and carriage returns) with spaces:

```
<xs:element name="address">
    <xs:simpleType>
        <xs:restriction base="xs:string">
            <xs:whiteSpace value="replace"/>
        </xs:restriction>
    </xs:simpleType>
</xs:element>
```

Restrictions on a Set of Values

The whiteSpace constraint is set to "collapse", which means that the XML processor WILL REMOVE all white space characters (line feeds, tabs, spaces, carriage returns are replaced with spaces, leading and trailing spaces are removed, and multiple spaces are reduced to a single space):

```
<xs:element name="address">
    <xs:simpleType>
        <xs:restriction base="xs:string">
            <xs:whiteSpace value="collapse"/>
        </xs:restriction>
    </xs:simpleType>
</xs:element>
```

Restrictions on Values

```
<xs:element    name="age">
 <xs:simpleType>
  <xs:restriction  base="xs:integer">
   <xs:minInclusive  value="0"/>
   <xs:maxInclusive  value="120"/>
  </xs:restriction>
 </xs:simpleType>
</xs:element>


<xs:element   name="password">
 <xs:simpleType>
  <xs:restriction  base="xs:string">
   <xs:pattern     value="[a-zA-Z0-9]{8}"/>
  </xs:restriction>
 </xs:simpleType>
</xs:element>
```

Restrictions on Values

```
<xs:element name="letter">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:pattern value="([a-z])*"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
<xs:element name="letter">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:pattern value="([a-z][A-Z])+"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

Restrictions on Values

```
<xs:element name="gender">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:pattern value="male|female"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>

<xs:element name="password">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:length value="8"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

Restrictions on Values

```
<xs:element name="password">
        <xs:simpleType>
                <xs:restriction base="xs:string">
                        <xs:minLength value="5"/>
                        <xs:maxLength value="8"/>
                </xs:restriction>
        </xs:simpleType>
</xs:element>
```

Restrictions for Datatypes

Constraint	Description
enumeration	Defines a list of acceptable values
fractionDigits	Specifies the maximum number of decimal places allowed. Must be equal to or greater than zero
length	Specifies the exact number of characters or list items allowed. Must be equal to or greater than zero
maxExclusive	Specifies the upper bounds for numeric values (the value must be less than this value)
maxInclusive	Specifies the upper bounds for numeric values (the value must be less than or equal to this value)
maxLength	Specifies the maximum number of characters or list items allowed. Must be equal to or greater than zero

Restrictions for Data types (..continued)

Constraint	Description
minExclusive	Specifies the lower bounds for numeric values (the value must be greater than this value)
minInclusive	Specifies the lower bounds for numeric values (the value must be greater than or equal to this value)
minLength	Specifies the minimum number of characters or list items allowed. Must be equal to or greater than zero
pattern	Defines the exact sequence of characters that are acceptable
totalDigits	Specifies the exact number of digits allowed. Must be greater than zero
whiteSpace	Specifies how white space (line feeds, tabs, spaces, and carriage returns) is handled

Simple Type

There are two types in schema file: **Simple Type** and **Complex Type**. **Simple Type** means that this type doesn't have attribute and elements except for the character data.

XSD has over 40 built-in Simple Types such as xsd:string, xsd:integer, etc. and custom-made data types of Simple Type as well.

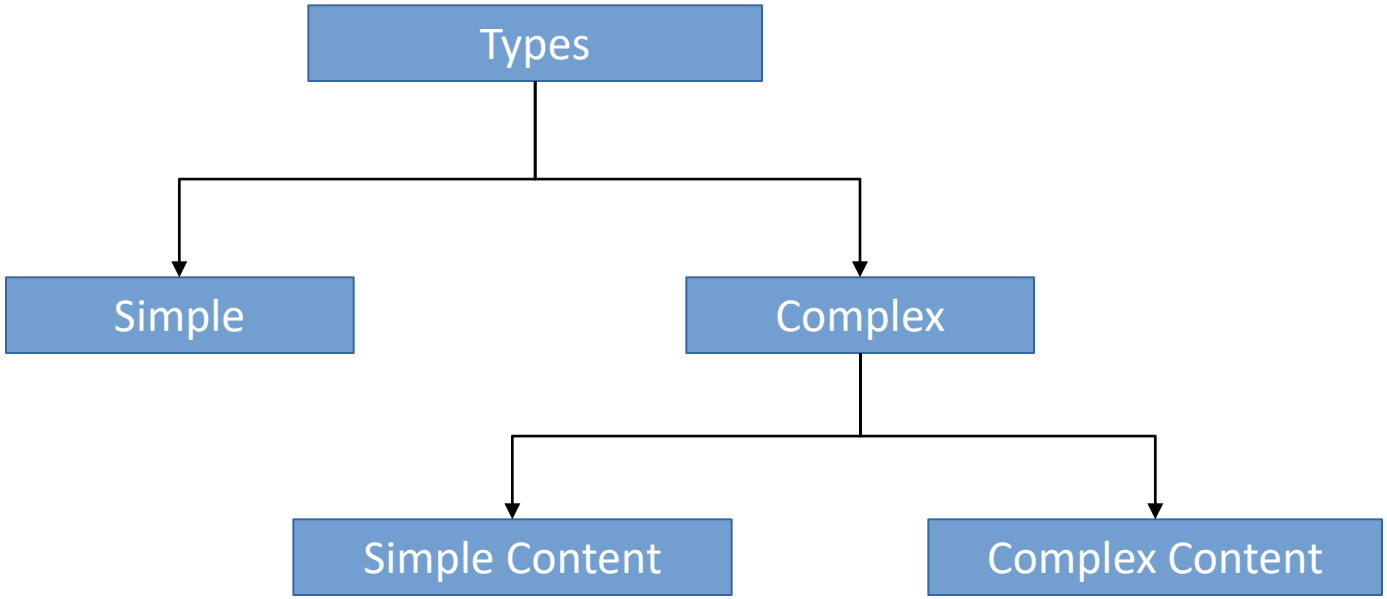
All custom-made Simple Type are the restriction, list of the built-in types or other custom-made simple types.

Complex Type

Contrary to **Simple Type, Complex Type** must either have attribute or elements or both of them.

Simple Content and Complex Content are about how the content of a Complex Type look like. Thus they have NOTHING to do with Simple Type.

Types in Schema



Allow attributes,
Allow characters

Allow attributes,
Allow child elements

XSD Complex Elements

There are four kinds of complex elements:

1) **empty** elements (e.g.
</br> or </br>)

2) **contain only other elements**

3) **contain only text with attribute(s)**

4) **contain both other elements and text**

- **Note:** Each of these elements may contain attributes as well!

Examples of Complex Elements

1) empty:

```
<product pid="1345"/>
```

2) contains only other elements:

```
<employee>
    <firstname>John</firstname>
    <lastname>Smith</lastname>
</employee>
```

3) contains only text with attribute:

```
<food   type="dessert">Ice cream</food>
```

4) contains both elements and text:

```
<description>
It happened on <date lang="norwegian">03.03.99</date> ....
</description>
```

```
<xs:element    name="employee">
    <xs:complexType>
     <xs:complexContent>
      <xs:restriction    base="anyType">
        <xs:sequence>
          <xs:element   name="firstname"   type="xs:string"/>
          <xs:element   name="lastname"    type="xs:string"/>
        </xs:sequence>
      </xs:restriction>
     </xs:complexContent>
    </xs:complexType>
</xs:element>
```

Abbreviated Form

```
<xs:element   name="employee">
   <xs:complexType>
      <xs:sequence>
         <xs:element   name="firstname"   type="xs:string"/>
         <xs:element   name="lastname"    type="xs:string"/>
      </xs:sequence>
   </xs:complexType>
</xs:element>
```

- Note that the child elements, "firstname" and "lastname", are surrounded by the <sequence> indicator.
- This means that the child elements must appear in the same order as they are declared.

Type Attribute

- The "employee" element can have a **type** attribute that refers to the name of the complex type to use:

```
<xs:element name="employee" 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>
```

- Several elements can refer to the same complex type, like this:

```
<xs:element name="employee" type="personinfo"/>
<xs:element name="student"    type="personinfo"/>
<xs:element name="member"    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>
```

- You can also base a complex element on an existing complex element

```
<xs:element name="employee" type="fullpersoninfo"/>

<xs:complexType    name="personinfo">
 <xs:sequence>
  <xs:element    name="firstname"    type="xs:string"/>
  <xs:element    name="lastname"     type="xs:string"/>
 </xs:sequence>
</xs:complexType>

<xs:complexType    name="fullpersoninfo">
 <xs:complexContent>
  <xs:extension   base="personinfo">
   <xs:sequence>
    <xs:element name="address"      type="xs:string"/>
    <xs:element name="city"         type="xs:string"/>
    <xs:element name="country"      type="xs:string"/>
   </xs:sequence>
  </xs:extension>
 </xs:complexContent>
</xs:complexType>
```

Complex Empty Elements

An empty XML element:

```
<product prodid="1345" />
```

```
<xs:element name="product">
    <xs:complexType>
        <xs:attribute name="prodid" type="xs:positiveInteger"/>
    </xs:complexType>
</xs:element>
```

Complex Empty Elements

An empty XML element:

<empty />

```
<xs:element name="empty">
    <xs:complexType ></xs:complexType>
</xs:element>
```

Defining Custom Type

```
<xs: element name="product" type="prodtype"/>


<xs: complexType name="prodtype">

    <xs: attribute   name="prodid"
                     type="xs: string" />

</xs:complexType>
```

Complex Types Containing Elements Only

An "elements-only" complex type contains an element that contains only other elements.

Example 1

```
<xs:element name="person">
  <xs:complexType>
        <xs:sequence>
              <xs:element name="firstname" type="xs:string"/>
              <xs:element name="lastname" type="xs:string"/>
        </xs:sequence>
  </xs:complexType>
</xs:element>
```

Complex Types Containing Elements Only

An "elements-only" complex type contains an element that contains only other elements.

Example 2

```
<xs:element name="person" type="persontype"/>

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

Complex Text-Only Elements

A complex text-only element can contain text and attributes.

This type contains only simple content (text and attributes), therefore we add a simpleContent element around the content.

When using simple content, you must define an **extension OR a restriction** within the simpleContent element, like this:

```
<xs:element name="somename">
  <xs:complexType>
        <xs:simpleContent>
              <xs:extension base="basetype">
                ....
              </xs:extension>
        </xs:simpleContent>
  </xs:complexType>
</xs:element>
```

Complex Text-Only Elements

OR

```
<xs:element name="somename">
  <xs:complexType>
        <xs:simpleContent>
              <xs:restriction base="basetype">
               ....
              </xs:restriction>
        </xs:simpleContent>
  </xs:complexType>
</xs:element>
```

Complex Text-Only Elements (Example 1)

```
<shoesize   country="france">35</shoesize>

<xs:element    name="shoesize">
    <xs:complexType>
        <xs:simpleContent>
            <xs:extension base="xs:integer">
                <xs:attribute name="country" type="xs:string"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
</xs:element>
```

Complex Text-Only Elements (Example 2)

```
<xs:element    name="shoesize"   type= "shoetype">

<xs:complexType     name="shoetype">
    <xs:simpleContent>
        <xs:extension base="xs:integer">
            <xs:attribute name="country" type="xs:string"/>
        </xs:extension>
    </xs:simpleContent>
</xs:complexType>
```

Complex Types with Mixed Content

```
<letter>
  Dear Mr.<name>John Smith</name>.
  Your order <orderid>1032</orderid>
  will be shipped on <shipdate>2001-07-13</shipdate>.
</letter>
```

Complex Types with Mixed Content

```
<xs:element name="letter">
  <xs:complexType mixed="true">
      <xs:sequence>
          <xs:element name="name"     type="xs:string"/>
          <xs:element name="orderid"  type="xs:positiveInteger"/>
          <xs:element name="shipdate" type="xs:date"/>
      </xs:sequence>
  </xs:complexType>
</xs:element>
```

OR

```
<xs:element name="letter" type="lettertype"/>

<xs:complexType name="lettertype" mixed="true">
  <xs:sequence>
        <xs:element name="name"       type="xs:string"/>
        <xs:element name="orderid"    type="xs:positiveInteger"/>
        <xs:element name="shipdate"   type="xs:date"/>
  </xs:sequence>
</xs:complexType>
```

Indicators

We can control HOW elements are to be used in documents with indicators.

Order indicators:

- All
- Choice
- Sequence

Occurrence indicators:

- maxOccurs
- minOccurs

Group indicators:

- Group name
- attributeGroup name

Order Indicators (All)

- The <all> declaration says that the elements can appear in any order, with each child element occurring zero or one time.
- The <all> declaration must be the only content model declaration that appears as a child of a <complexType> defi nition.
- The <all> declaration can contain only <element> declarations as its children.
- It is not permitted to contain <sequence>, <choice>, or <group> declarations.
- The <all> declaration’s children may appear once each in the instance document.
- Within the <all> declaration, the values for minOccurs for maxOccurs are limited to 0 or 1.

Order Indicators (All)

```
<xs:element name="person">
  <xs:complexType>
    <xs:all>
      <xs:element name="firstname" type="xs:string"/>
      <xs:element name="lastname" type="xs:string"/>
    </xs:all>
  </xs:complexType>
</xs:element>
```

Order Indicators (Choice)

The <choice> indicator specifies that Only one of the elements in the list may appear:

```
<xs:element name="person">
  <xs:complexType>
    <xs:choice>
      <xs:element name="employee" type="employee"/>
      <xs:element name="member" type="member"/>
    </xs:choice>
  </xs:complexType>
</xs:element>
```

Occurrence Indicators

maxOccurs Indicator

specifies the maximum number of times an element can occur:

```
<xs:element name="person">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="full_name" type="xs:string"/>
      <xs:element name="child_name" type="xs:string"
maxOccurs="5"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Tip: To allow an element to appear an unlimited number of times, use the **maxOccurs="unbounded"** statement:

Occurrence Indicators

minOccurs Indicator

specifies the minimum number of times an element can occur:

```
<xs:element name="person">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="full_name" type="xs:string"/>
            <xs:element name="child_name" type="xs:string"
            maxOccurs="5" minOccurs="0"/>
        </xs:sequence>
 </xs:complexType>
</xs:element>
```

Myfamily.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<persons xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="family.xsd">
<person>
        <full_name>Hege Refsnes</full_name>
        <child_name>Cecilie</child_name>
</person>
<person>
        <full_name>Tove Refsnes</full_name>
        <child_name>Hege</child_name>
        <child_name>Stale</child_name>
        <child_name>Jim</child_name>
        <child_name>Borge</child_name>
</person>
<person>
      <full_name>Stale Refsnes</full_name>
</person>
</persons>
```

family.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema   xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
<xs:element name="persons">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="person" maxOccurs="unbounded">
    <xs:complexType>
     <xs:sequence>
      <xs:element name="full_name" type="xs:string"/>
      <xs:element name="child_name" type="xs:string"
      minOccurs="0" maxOccurs="5"/>
     </xs:sequence>
    </xs:complexType>
   </xs:element>
  </xs:sequence>
 </xs:complexType>
</xs:element>
</xs:schema>
```

Group Indicators

Group indicators are used to define related sets of elements.

1) Element Groups

You must define an all, choice, or sequence element inside the group declaration.

```
<xs:group name="persongroup">
  <xs:sequence>
    <xs:element name="firstname" type="xs:string"/>
    <xs:element name="lastname" type="xs:string"/>
    <xs:element name="birthday" type="xs:date"/>
  </xs:sequence>
</xs:group>
```

Groups can only contain content models or an annotation, (i.e. their immediate children can only be one of the 3: <xs:all />, <xs:choice />, <xs:sequence />, or a documentation element). See https://www.w3.org/TR/xmlschema11-1/#cModel_Group_Definitions under section 3.7.2:

```
<group
  id = ID
  maxOccurs = (nonNegativeInteger | unbounded)  : 1
  minOccurs = nonNegativeInteger : 1
  name = NCName
  ref = QName
  {any attributes with non-schema namespace . . .}>
  Content: (annotation?, (all | choice | sequence)?)
</group>
```

Complex types, behave very similarly, as they can also have the exact same content models that groups have, but they also can define <xs:attributes>. You can even have complex types that define nothing but attributes, something you cannot do with groups.

Group Indicators

You can reference it in another definition, like this:

```
<xs:group   name="persongroup">
  <xs:sequence>
    <xs:element name="firstname" type="xs:string"/>
    <xs:element name="lastname" type="xs:string"/>
    <xs:element name="birthday" type="xs:date"/>
  </xs:sequence>
</xs:group>

<xs:element name="person" type="personinfo"/>

<xs:complexType name="personinfo">
  <xs:sequence>
    <xs:group    ref="persongroup"/>
    <xs:element   name="country"    type="xs:string"/>
  </xs:sequence>
</xs:complexType>
```

Group Indicators

2) Attribute Groups

```
<xs:attributeGroup    name="personattrgroup">
  <xs:attribute name="firstname" type="xs:string"/>
  <xs:attribute name="lastname" type="xs:string"/>
  <xs:attribute name="birthday" type="xs:date"/>
</xs:attributeGroup>
```

Group Indicators

You can reference it in another definition, like this:

```
<xs:attributeGroup name="personattrgroup">
    <xs:attribute    name="firstname" type="xs:string"/>
    <xs:attribute    name="lastname"  type="xs:string"/>
    <xs:attribute    name="birthday"  type="xs:date"/>
</xs:attributeGroup>

<xs:element name="person">
 <xs:complexType>
       <xs:attributeGroup ref="personattrgroup"/>
 </xs:complexType>
</xs:element>
```

The <any> Element

The <any> element enables us to extend the XML document with elements not specified by the schema.

```
<xs:element name="person">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="firstname" type="xs:string"/>
      <xs:element name="lastname" type="xs:string"/>
      <xs:any minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

children.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.w3schools.com"
xmlns="http://www.w3schools.com"
elementFormDefault="qualified">

<xs:element name="children">
	<xs:complexType>
		<xs:sequence>
			<xs:element name="childname" type="xs:string"
			maxOccurs="unbounded"/>
		</xs:sequence>
	</xs:complexType>
</xs:element>

</xs:schema>
```

Myfamily.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<persons xmlns="http://www.microsoft.com"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.microsoft.com family.xsd
http://www.w3schools.com children.xsd">

<person>
    <firstname>Hege</firstname>
    <lastname>Refsnes</lastname>
    <children>
        <childname>Cecilie</childname>
    </children>
</person>
<person>
    <firstname>Stale</firstname>
    <lastname>Refsnes</lastname>
</person>
</persons>
```

The <anyAttribute> Element

The <anyAttribute> element enables us to extend the XML document with attributes not specified by the schema.

Fragment from family.xsd

```
<xs:element name="person">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="firstname" type="xs:string"/>
      <xs:element name="lastname" type="xs:string"/>
    </xs:sequence>
    <xs:anyAttribute/>
  </xs:complexType>
</xs:element>
```

attribute.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.w3schools.com"
xmlns="http://www.w3schools.com"
elementFormDefault="qualified">

<xs:attribute name="gender">
        <xs:simpleType>
                <xs:restriction base="xs:string">
                        <xs:pattern value="male|female"/>
                </xs:restriction>
        </xs:simpleType>
</xs:attribute>

</xs:schema>
```

Myfamily.xml

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

<persons xmlns="http://www.microsoft.com"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:SchemaLocation="http://www.microsoft.com family.xsd
http://www.w3schools.com attribute.xsd">

<person gender="female">
  <firstname>Hege</firstname>
  <lastname>Refsnes</lastname>
</person>

<person gender="male">
  <firstname>Stale</firstname>
  <lastname>Refsnes</lastname>
</person>

</persons>
```

Element Substitution

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

<xs:complexType name="custinfo">
  <xs:sequence>
    <xs:element ref="name"/>
  </xs:sequence>
</xs:complexType>

<xs:element name="customer" type="custinfo"/>
<xs:element name="kunde" substitutionGroup="customer"/>
```

XML document may contain

```
<customer>
  <name>John Smith</name>
</customer>
```

or

```
<kunde>
  <navn>John Smith</navn>
</kunde>
```

Blocking Element Substitution

<xs:element name="name" type="xs:string" **block="substitution**"/>

shiporder.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<shiporder orderid="889923"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="shiporder.xsd">
  <orderperson>John Smith</orderperson>
  <shipto>
    <name>Ola Nordmann</name>
    <address>Langgt 23</address>
    <city>4000 Stavanger</city>
    <country>Norway</country>
  </shipto>
  <item>
    <title>Empire Burlesque</title>
    <note>Special Edition</note>
    <quantity>1</quantity>
    <price>10.90</price>
  </item>
  <item>
    <title>Hide your heart</title>
    <quantity>1</quantity>
    <price>9.90</price>  </item></shiporder>
```

shiporder.xsd

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

<xs:element name="shiporder">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="orderperson" type="xs:string"/>
      <xs:element name="shipto">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="name" type="xs:string"/>
            <xs:element name="address" type="xs:string"/>
            <xs:element name="city" type="xs:string"/>
            <xs:element name="country" type="xs:string"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
```

shiporder.xsd (..continued)

```
    <xs:element name="item" maxOccurs="unbounded">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="title" type="xs:string"/>
          <xs:element name="note" type="xs:string" minOccurs="0"/>
          <xs:element name="quantity" type="xs:positiveInteger"/>
          <xs:element name="price" type="xs:decimal"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
  <xs:attribute name="orderid" type="xs:string" use="required"/>
 </xs:complexType>
</xs:element>

</xs:schema>
```

shiporder2.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<shiporder orderid="889923"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="shiporder.xsd">
  <orderperson>John Smith</orderperson>
  <shipto>
    <name>Ola Nordmann</name>
    <address>Langgt 23</address>
    <city>4000 Stavanger</city>
    <country>Norway</country>
  </shipto>
  <item itemid = "a1">
    <title>Empire Burlesque</title>
    <note>Special Edition</note>
    <quantity>1</quantity>
    <price>10.90</price>
  </item>
  <item itemid = "a2">
    <title>Hide your heart</title>
    <quantity>1</quantity>
    <price>9.90</price>  </item></shiporder>
```

shiporder2.xsd

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

<!-- definition of simple elements -->
<xs:element   name="orderperson" type="xs:string"/>
<xs:element   name="name"        type="xs:string"/>
<xs:element   name="address"     type="xs:string"/>
<xs:element   name="city"        type="xs:string"/>
<xs:element   name="country"     type="xs:string"/>
<xs:element   name="title"       type="xs:string"/>
<xs:element   name="note"        type="xs:string"/>
<xs:element   name="quantity"    type="xs:positiveInteger"/>
<xs:element   name="price"       type="xs:decimal"/>

<xs:attribute  name="orderid"    type="xs:string"/>
<xs:attribute  name="itemid"     type="xs:ID"/>
```

```
<!-- definition of complex elements -->
<xs:element name="shipto">
  <xs:complexType>
    <xs:sequence>
      <xs:element     ref="name"/>
      <xs:element     ref="address"/>
      <xs:element     ref="city"/>
      <xs:element     ref="country"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

```
<xs:element name="item">
  <xs:complexType>
    <xs:sequence>
      <xs:element      ref="title"/>
      <xs:element     ref="note"        minOccurs="0"/>
      <xs:element     ref="quantity"/>
      <xs:element     ref="price"/>
    </xs:sequence>
    <xs:attribute          ref="itemid" use="required"/>
  </xs:complexType>
</xs:element>
```

```
<xs:element name="shiporder">
  <xs:complexType>
    <xs:sequence>
      <xs:element  ref="orderperson"/>
      <xs:element  ref="shipto"/>
      <xs:element  ref="item"     maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute   ref="orderid" use="required"/>
  </xs:complexType>
</xs:element>

</xs:schema>
```

shiporder3.xsd

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema
xmlns:xs="http://www.w3.org/2001/XMLSchema">

<xs:simpleType name="stringtype">
  <xs:restriction base="xs:string"/>
</xs:simpleType>

<xs:simpleType name="inttype">
  <xs:restriction base="xs:positiveInteger"/>
</xs:simpleType>

<xs:simpleType name="dectype">
  <xs:restriction base="xs:decimal"/>
</xs:simpleType>
```

shiporder3.xsd (..continued)

```
<xs:simpleType name="orderidtype">
  <xs:restriction base="xs:string">
    <xs:pattern  value="[0-9]{6}"/>
  </xs:restriction>
</xs:simpleType>

<xs:complexType name="shiptotype">
  <xs:sequence>
    <xs:element name="name"          type="stringtype"/>
    <xs:element name="address"       type="stringtype"/>
    <xs:element name="city"          type="stringtype"/>
    <xs:element name="country"       type="stringtype"/>
  </xs:sequence>
</xs:complexType>
```

shiporder3.xsd (..continued)

```
<xs:complexType name="itemtype">
  <xs:sequence>
    <xs:element name="title"            type="stringtype"/>
    <xs:element name="note" type="stringtype"  minOccurs="0"/>
    <xs:element name="quantity"      type="inttype"/>
    <xs:element name="price"           type="dectype"/>
  </xs:sequence>
</xs:complexType>
```

shiporder3.xsd (..continued)

```
<xs:complexType name="shipordertype">
  <xs:sequence>
    <xs:element name="orderperson"     type="stringtype"/>
    <xs:element name="shipto"          type="shiptotype"/>
    <xs:element name="item"    type="itemtype" maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="orderid"       type="orderidtype"    use="required"/>
</xs:complexType>

<xs:element name="shiporder"       type="shipordertype"/>

</xs:schema>
```