XML Schema is commonly known as XML Schema Definition \(XSD\). It is used to describe and validate the structure and the content of XML data. XML schema defines the elements, attributes and data types. Schema element supports Namespaces. It is similar to a database schema that describes the data in a database.

**Syntax**

You need to declare a schema in your XML document as follows:

```xml
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
</xs:schema>
```

**Example**

The following example shows how to use schema:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="contact">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="name" type="xs:string" />
        <xs:element name="company" type="xs:string" />
        <xs:element name="phone" type="xs:int" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

The basic idea behind XML Schemas is that they describe the legitimate format that an XML document can take.

**Elements**

As we saw in the [XML - Elements](http://www.tutorialspoint.com/xml/xml_elements.htm) chapter, elements are the building blocks of XML document. An element can be defined within an XSD as follows:

```xml
<xs:element name="x" type="y"/>
```

**Definition Types**

You can define XML schema elements in following ways:

**Simple Type** - Simple type element is used only in the context of the text. Some of predefined simple types are: `xs:integer`, `xs:boolean`, `xs:string`, `xs:date`. For example:

```xml
<xs:element name="phone_number" type="xs:int" />
```

**Complex Type** - A complex type is a container for other element definitions. This allows you to specify which child elements an element can contain and to provide some structure within your XML documents. For example:

```xml
<xs:element name="Address">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="name" type="xs:string" />
      <xs:element name="company" type="xs:string" />
      <xs:element name="phone" type="xs:int" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
```
In the above example, Address element consists of child elements. This is a container for other <xs:element> definitions, that allows to build a simple hierarchy of elements in the XML document.

**Global Types** - With global type, you can define a single type in your document, which can be used by all other references. For example, suppose you want to generalize the person and company for different addresses of the company. In such case, you can define a general type as below:

```xml
<xs:element name="AddressType">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="name" type="xs:string" />
      <xs:element name="company" type="xs:string" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Now let us use this type in our example as below:

```xml
<xs:element name="Address1">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="address" type="AddressType" />
      <xs:element name="phone1" type="xs:int" />
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="Address2">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="address" type="AddressType" />
      <xs:element name="phone2" type="xs:int" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Instead of having to define the name and the company twice (once for Address1 and once for Address2), we now have a single definition. This makes maintenance simpler, i.e., if you decide to add "Postcode" elements to the address, you need to add them at just one place.

**Attributes**

Attributes in XSD provide extra information within an element. Attributes have name and type property as shown below:

```xml
<xs:attribute name="x" type="y"/>
```

Loading [MathJax]/jax/output/HTML-CSS/fonts/TeX/fontdata.js