# XML : Anatomy of XML Document

1)Syntactical rules of well form xml.

Ans)

**A Well-formed XML document:**

- A Well-formed XML document simply includes markup pages with descriptive tags
- A Well-formed XML does not need a DTD, but should conform to xml syntax
- If all tags are correctly formed and follow XML guidelines, then the document is a well-formed XML

**Syntax Rules for XML:**

An XML document

- Is case sensitive
- Has a single root element
- Has all matching tags
- XML Elements should be properly nested
- All attributes are quoted
- White spaces are not ignored
- May or may not have a (DTD) document Type Description to describe the document.

2)Rules to form element name.

Ans)

**Naming Rules:**

- A name consists of at least one letter : a to z or A-Z.
- If the name consists of more than one character, then it may start with an underscore ( _ ) or a colon (:)
- The initial letter can be followed by one or more letters, digits, hyphens, underscores, or full stops.

3)How to check whether an element contains mixed content.

Ans)

**XSD Mixed Content:**

- A mixed complex type element can contain attributes, elements, and text.

**Complex Types with Mixed Content**

- An XML element, "letter", that contains both text and other elements:

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

The following schema declares the "letter" element:

```
<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>
```

4)Compare dtd and xsd.

Ans)

**Drawbacks of DTD:**

- Use of non-XML syntax
- No support for data typing
- Non-extensibility

**Advantages of XML Schemas:**

- Support data types
- Use XML syntax
- Secure data communication
- Are extensible
- Well-formed is not enough

5)How to form complex type.

Ans)

**Complex Element:**

A complex element is an XML element that contains other elements and/or attributes.

There are four kinds of complex elements:

- Empty elements
- Elements that contain only other elements
- Elements that contain both – other elements and text
- Elements that contain text

Examples of Complex XML Elements:

Example 1:

Consider a complex XML element, “product”, which is empty:

- <product pid="1345"/>

It can be declared as:

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

# XML : Anatomy of XML Document

6)Importance of <xs:sequence>, <xs:all> and <xs:choice>.
Ans)
**All Indicator:**

- The <all> indicator specifies, by default, that the child elements can appear in any order and that each child element must occur once and only once.

**Choice Indicator:**

- The <choice> indicator specifies that either one child element or another can occur.

**Sequence Indicator:**

- The <sequence> indicator specifies that the child elements must appear in a specific order.

7)How to apply SimpleType restrictions such as : pattern, minInclusive, maxInclusive,
Ans)
Restrictions on XSD Elements:

**Restrictions on Content:**
When an XML element or attribute has a datatype associated with , it puts a restriction on the element's or attribute's content.

**Restrictions on Values:**
The example in the above slide defines an element called "Quantity" with a restriction. The value of book "Quantity" cannot be lower than 0 or greater than 500.

- MinInclusive Specifies the lower bounds for numeric values (the value must be greater than or equal to this value)
- MaxInclusive Specifies the upper bounds for numeric values (the value must be less than or equal to this value)

**Example:**

```
<xs:element name="Quantity">
    <xs:simpleType>
        <xs:restriction base="xs:integer">
            <xs:minInclusive value="0" />
            <xs:maxInclusive value="500"/>
        </xs:restriction>
    </xs:simpleType>
</xs:element>
```

**Restrictions on Series of Values**

- To limit the content of an XML elemnt to define a series of numbers or letters that can be used, we can use the pattern constraint.

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

```
        <xs:simpleType>
            <xs:restriction base="xs:string">
                <xs:pattern value="[a-z]" / >
            </xs:restriction>
        </xs:simpleType>
    </xs:element>
```

- The only acceptable value is ONE of the LOWERCASE letters from a to z.

8)Attribute definition in xsd with its usage.

Ans)

- Defining an Attribute:
  < xs : attribute  name="AuthorID" type="xs:string" />
  (where "AuthorID" is the name of the attribute and "xs:string" specifies the data type of the attribute )
- Creating optional and Required Attributes:
  < xs : attribute name="btype" type="xs:string" use="required" />
- Attributes are optional by default.