

XML Syntax Rules

- **Declaration:** Placed at the very first line of the document

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
```
- **Single Root Element:** Every valid XML document must contain exactly one root element.
- **Closing Tags & Case Sensitivity:** All opening tags must have matching closing tags. XML is case-sensitive. Self-closing tags use `<tag/>`.
- **Attributes:** Attribute values must always be enclosed in single (') or double (") quotes.
- **Proper Nesting:** Elements must nest properly without overlapping (`<b><i>text</i></b>`).
- **Comments:** `<!-- Comment text -->`

Predefined Entities & CDATA

Special characters must be escaped or wrapped in CDATA blocks to prevent parser errors.

- **Predefined Entities:**
 - `<`; `<` (Less than)
 - `>`; `>` (Greater than)
 - `&`; `&` (Ampersand)
 - `'`; `'` (Apostrophe)
 - `"`; `"` (Quote)
- **CDATA Section:** Preserves raw text without XML parsing

```
<![CDATA[
    function match(a, b) { return a < b
    && b > 0; }
]]>
```

Namespaces

Avoid element name conflicts by associating elements with a URI.

- **Prefixed Namespace Declaration:**

```
<root xmlns:h="http://www.w3.org/TR/html4/"
      xmlns:f="https://www.w3schools.com/furniture">
  <h:table><h:tr><h:td>Apples</h:td></h:tr></h:table>
  <f:table><f:name>Coffee Table</f:name></f:table>
</root>
```

- **Default Namespace:**

```
<table xmlns="http://www.w3.org/TR/html4/">
  <tr><td>Apples</td></tr>
</table>
```

XPath Expressions

Query and navigate nodes and attributes in an XML document.

Path Navigation

- `/`: Select from root node
- `//`: Select matching nodes anywhere in document
- `.`: Current node
- `..`: Parent of current node
- `@`: Select attributes (e.g. `//@id`)
- `*`: Match any element node
- `@*`: Match any attribute node

Predicates & Functions

- `/bookstore/book[1]`: Select first book element
- `/bookstore/book[last()]`: Select last book element
- `/bookstore/book[price>35.00]`: Select books where price > 35
- `//title[@lang='en']`: Select title elements with lang='en'
- `//book[contains(title, 'XML')]`: Select books containing 'XML' in title
- `count(//book)`: Count number of book nodes
- `//title/text()`: Extract text content of title

XPath Axes

- `ancestor::`: All ancestors of current node
- `child::`: Children of current node (default)
- `descendant::`: All descendants of current node
- `parent::`: Parent of current node
- `following-sibling::`: All siblings after current node
- `preceding-sibling::`: All siblings before current node

Validation: DTD & XSD

DTD (Document Type Definition)

Traditional structural schema definition.

```
<!DOCTYPE note [
  <!ELEMENT note (to,from,heading,body)>
  <!ELEMENT to (#PCDATA)>
  <!ELEMENT from (#PCDATA)>
  <!ELEMENT heading (#PCDATA)>
  <!ELEMENT body (#PCDATA)>
  <!ATTLIST note id CDATA #REQUIRED>
]>
```

XSD (XML Schema Definition)

Modern typed schema definition using XML syntax.

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="book">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="title"
          type="xs:string"/>
        <xs:element name="price"
          type="xs:decimal"/>
        <xs:element name="publishDate"
          type="xs:date"/>
      </xs:sequence>
      <xs:attribute name="id"
        type="xs:integer" use="required"/>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

- **Built-in XSD Types:** `xs:string`, `xs:decimal`, `xs:integer`, `xs:boolean`, `xs:date`, `xs:time`
- **Multiplicity:**
`minOccurs="0", maxOccurs="unbounded"`

XSLT Transformations

Transform XML documents into HTML, plain text, or alternative XML.

```
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">
    <html>
      <body>
```

```
      <h2>Book Catalog</h2>
      <table border="1">
        <tr>
          <th>Title</th>
          <th>Price</th>
          <xsl:for-each select="catalog/book">
            <xsl:if test="price > 10">
              <tr>
                <td><xsl:value-of
                  select="title"/></td>
                <td><xsl:value-of
                  select="price"/></td>
              </tr>
            </xsl:if>
          </xsl:for-each>
        </table>
      </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```

CLI Tools & Programming APIs

Command Line Tools

- `xmllint --format input.xml`: Format and pretty-print XML
- `xmllint --schema schema.xsd --noout file.xml`: Validate against XSD
- `xmllstarlet sel -t -v "//book/title" input.xml`: Extract values via XPath
- `xmllstarlet ed -u "//book/price" -v "29.99" input.xml`: Modify XML node

Python (xml.etree.ElementTree)

```
import xml.etree.ElementTree as ET
```

```
tree = ET.parse('data.xml')
root = tree.getroot()
```

```
for book in root.findall('book'):
    title = book.find('title').text
    book_id = book.attrib.get('id')
```

```
new_elem = ET.SubElement(root, 'book',
  id='102')
```

Last updated: Last updated: 2026-09-21

```
tree.write('output.xml',  
encoding='utf-8', xml_declaration=True)
```