

XML 기본 문법 규칙 (Syntax Rules)

- 선언 (Declaration): XML 문서의 첫 번째 줄에 작성

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
```
- 단일 루트 요소 (Root Element): 모든 XML 문서는 반드시 단 하나의 최상위 루트 요소를 가져야 합니다.
- 태그 닫기 및 대소문자 구분: 모든 시작 태그는 반드시 종료 태그를 가져야 하며 대소문자를 구분합니다. 빈 태그는 `<tag/>`로 자가 종료합니다.
- 속성 (Attributes): 속성값은 반드시 큰따옴표(") 또는 작은따옴표(')로 감싸야 합니다.
- 올바른 중첩 (Nesting): 태그는 순서에 맞게 올바르게 중첩되어야 합니다 (`<b><i>text</i></b>`).
- 주석 (Comments): `<!-- 주석 내용 -->`

사전 정의된 엔티티 및 CDATA

XML 파서가 마크업으로 해석하지 않도록 특수 문자를 치환하거나 블록 처리합니다.

- 엔티티 참조 (Predefined Entities):
 - `<` : `<` (Less than)
 - `>` : `>` (Greater than)
 - `&` : `&` (Ampersand)
 - `'` : `'` (Apostrophe)
 - `"` : `"` (Quote)
- CDATA 섹션: 특수 문자가 많은 원시 텍스트를 파싱하지 않고 그대로 포함할 때 사용

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

네임스페이스 (Namespaces)

요소 이름 충돌을 방지하기 위해 URI와 접두사를 연결합니다.

- 접두사 네임스페이스 선언:

```
<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>
```

- 기본 네임스페이스 선언:

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

XPath 쿼리 문법 (XPath Expressions)

XML 문서 내에서 특정 노드나 값을 탐색하기 위한 경로 언어입니다.

기본 노드 선택 경로

- `/` : 루트 노드부터 선택
- `//` : 현재 노드의 위치와 상관없이 문서 전체에서 일치하는 노드 탐색
- `.` : 현재 노드 선택
- `..` : 현재 노드의 부모 노드 선택
- `@` : 속성(Attribute) 선택 (예: `//@id`)
- `*` : 모든 요소 노드 일치
- `*@` : 모든 속성 노드 일치

조건부 술어 (Predicates) 및 함수

- `/bookstore/book[1]` : 첫 번째 book 요소 선택
- `/bookstore/book[last()]` : 마지막 book 요소 선택
- `/bookstore/book[price>35.00]` : price가 35보다 큰 book 선택
- `//title[@lang='ko']` : lang 속성이 'ko'인 title 요소 선택
- `//book[contains(title, 'XML')]` : 제목에 'XML'이 포함된 book 선택
- `count(//book)` : book 요소의 총 개수 계산
- `//title/text()` : title 요소의 텍스트 노드 추출

주요 XPath 축 (Axes)

- `ancestor::` : 현재 노드의 모든 상위 노드
- `child::` : 현재 노드의 직계 자식 노드 (기본값)
- `descendant::` : 현재 노드의 모든 하위 노드
- `parent::` : 현재 노드의 부모 노드
- `following-sibling::` : 현재 노드 뒤에 나오는 모든 형제 노드
- `preceding-sibling::` : 현재 노드 앞에 나오는 모든 형제 노드

XML 유효성 검증 (DTD & XSD)

DTD (Document Type Definition)

문서 구조를 검증하는 고전적 스키마 정의.

```
<!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)

XML 문법 기반의 데이터 타입 지원 현대 스키마.

```
<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>
```

- 기본 XSD 타입: `xs:string`, `xs:decimal`, `xs:integer`, `xs:boolean`, `xs:date`, `xs:time`
- 요소 발생 횟수: `minOccurs="0"`, `maxOccurs="unbounded"`

XSLT 스타일시트 변환 (XSLT Basics)

XML 데이터를 HTML이나 다른 XML 문서로 변환하는 템플릿 언어.

```
<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>
      <td colspan="2"><th>Title</th><th>Price</th></tr>
      <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>
```

유용한 CLI 도구 및 프로그래밍 조작

터미널 CLI 도구

- `xmllint --format input.xml`: XML 들여쓰기 정렬 및 서식화
- `xmllint --schema schema.xsd --noout file.xml`: XSD 기반 유효성 검증
- `xmllstarlet sel -t -v "//book/title" input.xml`: XPath 값 추출
- `xmllstarlet ed -u "//book/price" -v "29.99" input.xml`: XML 노드 값 수정

Python (xml.etree.ElementTree)

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

```
# 파일 파싱 및 순회
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')
```

새 요소 추가 및 저장

Last updated: 최종 업데이트: 2026-09-21

```
new_elem = ET.SubElement(root, 'book',  
id='102')  
tree.write('output.xml',  
encoding='utf-8', xml_declaration=True)
```