Web Data Architectures & Web Security

Comprehensive Technical Notes: XML Fundamentals, JSON vs. XML, XML Schema & Transformation (XSLT/XPath), and Web Security Fundamentals

---

Module 1: XML Fundamentals & Document Structure

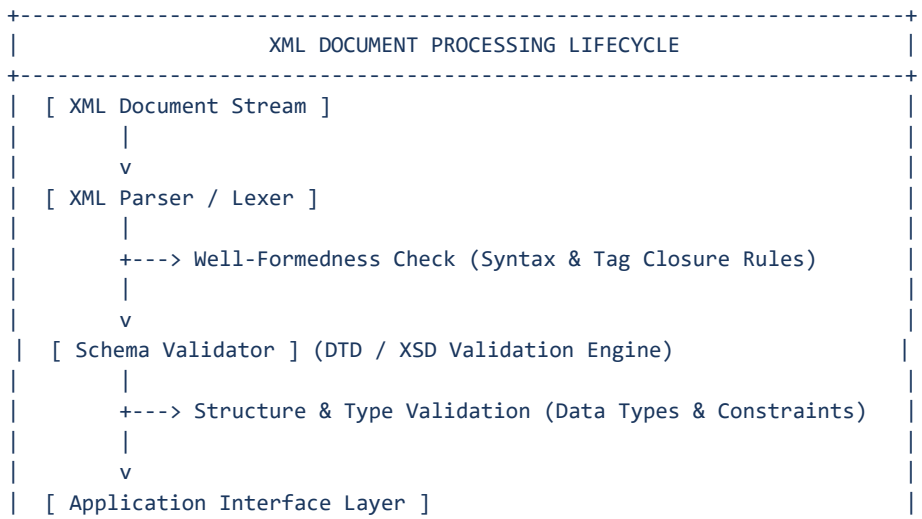
Extensible Markup Language (XML) is a self-describing, software- and hardware-independent markup language designed for storing, transporting, and structuring data across diverse distributed networks [cite: 1, 2].

1.1 XML Syntax, Rules, and Tree Structure

Every valid XML document forms an ordered, hierarchical tree structure beginning at a single root element and branching into child elements, attributes, and text nodes [cite: 1, 2]. Core syntactic rules include [cite: 1, 2]:

- **Prolog Declaration:** <?xml version="1.0" encoding="UTF-8"?> defines the metadata, XML version, and character encoding.
- **Single Root Element:** Every XML document must contain exactly one top-level root element enclosing all other elements.
- **Strict Tag Closure & Nesting:** All elements must have explicit opening and closing tags, strictly nested without overlapping tags.
- **Case Sensitivity & Attribute Quotes:** Element tags are case-sensitive (<Data> != <data>), and all attribute values must be enclosed in double or single quotes.

1.2 XML Document Processing Lifecycle Diagram



```
|       |                                                                  |
|       +---> DOM Parser (In-Memory Tree Construction)                    |
|       |                                                                  |
|       +---> SAX / StAX Parser (Event-Driven Streaming)                   |
+-------------------------------------------------------------------------+
```

> □ **DOM VS. SAX PARSING:** DOM (Document Object Model) loads the complete XML tree into RAM for random access, whereas SAX (Simple API for XML) streams events line-by-line, making SAX ideal for multi-gigabyte XML processing.

Module 2: Structural Comparison: JSON vs. XML

Both JSON (JavaScript Object Notation) and XML are lightweight data-interchange formats, but they cater to distinct architectural use cases in web applications and enterprise systems [cite: 1, 2].

2.1 Comparative Analysis Matrix

Dimension	JSON (JavaScript Object Notation)	XML (Extensible Markup Language)
Data Model	Key-Value pairs, Arrays, Objects	Hierarchical Tree (Elements, Attributes, Nodes)
Native Data Types	String, Number, Boolean, Array, Object, Null	Text / Untyped Strings (Requires XSD for types)
Human Readability	High (Concise, minimalist syntax)	Moderate (Verbose due to markup tags)
Schema Validation	JSON Schema	XSD (XML Schema Definition) / DTD
Extensibility & Namespaces	No native namespace support	Built-in XML Namespaces (xmlns)
Transformation Engine	JSONata, custom JS mappings	XSLT (Extensible Stylesheet Language)
Primary Application Domain	REST APIs, Web/Mobile Apps, Microservices	SOAP Web Services, Enterprise Messaging, Document Specs

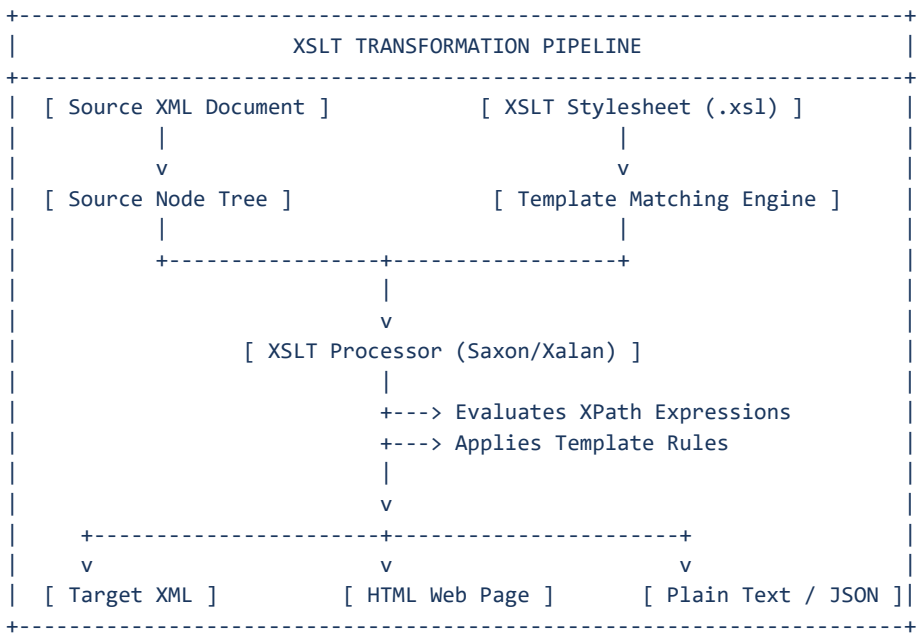
Module 3: XML Schema Definition (XSD) & Transformation (XSLT/XPath)

XML Schema Definition (XSD) defines the legal building blocks, elements, data types, and structural relationships of an XML document, while XSLT transforms XML into HTML, plain text, or alternative XML schemas [cite: 1, 2].

3.1 XSD Constraints, Data Types, and XPath Locators

- **Simple vs. Complex Types:** Simple types contain only text/value constraints (e.g., xs:string, xs:integer). Complex types contain child elements and attributes.
- **Occurrence Indicators:** minOccurs and maxOccurs define cardinality constraints for elements.
- **XPath Navigation:** Path expressions (e.g., /bookstore/book[price>35]/title) navigate XML node trees for retrieval and transformation.

3.2 Block Diagram: XSLT Transformation Pipeline



Module 4: Web Security Fundamentals & Threat Mitigation

Web security spans client-side browser protections, network transport layer security, API authentication protocols, and backend defense against injection attacks.

4.1 Core Vulnerabilities: OWASP Top Threats

- **Cross-Site Scripting (XSS):** Attacks where malicious scripts are injected into trusted web applications (Stored, Reflected, DOM-based XSS). Mitigated via HTML entity encoding and Content Security Policy (CSP).

- **Cross-Site Request Forgery (CSRF):** Forces an authenticated user's browser to send unauthorized HTTP requests. Mitigated using Anti-CSRF tokens and SameSite cookie attributes.
- **SQL & XML Injection (XXE):** Inadequate input sanitization allowing attackers to manipulate database queries or exploit external XML entity resolution to read local server files.
- **Broken Authentication & CORS:** Improper session management and misconfigured Cross-Origin Resource Sharing headers exposing protected API endpoints.

4.2 Block Diagram: End-to-End Web Application Defense Architecture



> **DEFENSE IN DEPTH:** Defense-in-depth requires layered protections: TLS at the transport layer, WAF at the perimeter, OAuth2/JWT at the API gateway, parameterized queries at the application layer, and disabled external entities in XML parsers.