INTRODUCTION TO XML

Extensible Markup Language (XML) is a markup language and file format for storing, transporting, and organizing data in a format that is both human-readable and machine-readable. It uses custom, self-descriptive tags to define the structure and meaning of data, unlike HTML, which uses predefined tags to display data.

Key Concepts

- **Self-descriptive:** XML documents use tags that describe the data they contain. For example, `<to>Bob</to>` clearly identifies "Bob" as the recipient.
- **Platform-independent:** Data stored in plain text XML format can be easily shared between different systems, applications, and programming languages without data loss.
- **Hierarchical structure:** XML documents form a tree-like structure, starting from a single root element that contains all other elements (known as child elements).
- **Separation of data and presentation:** XML focuses on the content and structure of data, leaving the presentation (how it looks) to other technologies like CSS or XSLT.
- **W3C Recommendation:** XML is an open standard developed and maintained by the World Wide Web Consortium (W3C).

Core Syntax Rules

A "well-formed" XML document must follow specific syntax rules:

- XML documents must have one unique root element that is the parent of all other elements.
- All XML elements must have a closing tag (e.g., `<message>...</message>`).
- XML tags are case-sensitive (e.g., `<Letter>` is different from `<letter>`).
- Elements must be properly nested (e.g., `<b><i>...</i></b>` is correct, `<b><i>...</b></i>` is not).
- Attribute values must always be quoted.
- Special characters like `<` and `&` must be replaced by entity references (e.g., `&lt;` and `&amp;`).

Primary Uses

XML is used in a wide variety of applications:

- **Data interchange:** It serves as a universal format for exchanging data between incompatible computer systems and applications, such as in web services (SOAP).
- **Configuration files:** Many applications, including Android mobile apps and Microsoft Office file formats (.docx, .xlsx), use XML to store settings and configuration data.
- **Document formatting:** It is used in publishing and other fields to structure documents for flexible output to different media (screens, print, etc.).
- **APIs:** It is often used in the implementation of APIs to transfer structured data between a client and a server.

XML is a software- and hardware-independent tool for storing and transporting data.

What is XML?

- XML stands for eXtensible Markup Language
- XML is a markup language much like HTML
- XML was designed to store and transport data
- XML was designed to be self-descriptive
- XML is a W3C Recommendation

XML Does Not DO Anything

Maybe it is a little hard to understand, but XML does not DO anything.

This note is a note to Tove from Jani, stored as XML:

```
<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

The XML above is quite self-descriptive:

- It has sender information
- It has receiver information
- It has a heading
- It has a message body

But still, the XML above does not DO anything. XML is just information wrapped in tags.

Someone must write a piece of software to send, receive, store, or display it:

Note

To: Tove

From: Jani

Reminder

Don't forget me this weekend!

The Difference Between XML and HTML

XML and HTML were designed with different goals:

- XML was designed to carry data - with focus on what data is
- HTML was designed to display data - with focus on how data looks
- XML tags are not predefined like HTML tags are

XML Does Not Use Predefined Tags

The XML language has no predefined tags.

The tags in the example above (like <to> and <from>) are not defined in any XML standard. These tags are "invented" by the author of the XML document.

HTML works with predefined tags like <p>, <h1>, <table>, etc.

With XML, the author must define both the tags and the document structure.

XML is Extensible

Most XML applications will work as expected even if new data is added (or removed).

Imagine an application designed to display the original version of note.xml (<to> <from> <heading> <body>).

Then imagine a newer version of note.xml with added <date> and <hour> elements, and a removed <heading>.

The way XML is constructed, older version of the application can still work:

```
<note>
  <date>2015-09-01</date>
  <hour>08:30</hour>
  <to>Tove</to>
  <from>Jani</from>
```

```
  <body>Don't forget me this weekend!</body>
</note>
```

Old Version

Note

To: Tove

From: Jani

Reminder

Don't forget me this weekend!

New Version

Note

To: Tove

From: Jani

Date: 2015-09-01 08:30

Don't forget me this weekend!

XML Simplifies Things

- XML simplifies data sharing
- XML simplifies data transport
- XML simplifies platform changes
- XML simplifies data availability

Many computer systems contain data in incompatible formats. Exchanging data between incompatible systems (or upgraded systems) is a time-consuming task for web developers. Large amounts of data must be converted, and incompatible data is often lost.

XML stores data in plain text format. This provides a software- and hardware-independent way of storing, transporting, and sharing data.

XML also makes it easier to expand or upgrade to new operating systems, new applications, or new browsers, without losing data.

With XML, data can be available to all kinds of "reading machines" like people, computers, voice machines, news feeds, etc.

XML is a W3C Recommendation

XML became a W3C Recommendation as early as in February 1998

XML BASICS

Extensible Markup Language (XML) is a markup language designed to store and transport data, making it readable by both humans and machines. Unlike HTML, which uses predefined tags to display data, XML allows authors to define their own tags to describe the data's structure and meaning.

Core Concepts

- **Purpose:** XML's primary goal is to carry data, focusing on "what the data is", rather than how it should look or be displayed.
- **Extensibility:** Users can create an unlimited number of custom tags to fit specific data needs, which is why it is "extensible".
- **Self-Descriptive:** The use of descriptive tags, such as `<to>`, `<from>`, and `<heading>`, makes the data largely understandable on its own.
- **Platform Independence:** XML stores data in a plain text format, providing a software- and hardware-independent way to store, transport, and share data across incompatible systems.
- **Tree Structure:** An XML document is structured as a single root element that contains all other elements, forming a hierarchical, tree-like structure.

Basic Syntax Rules

For an XML document to be considered "well-formed," it must adhere to a strict set of syntax rules:

- **Root Element:** Every XML document must have one and only one root element that is the parent of all other elements.
- **Closing Tags:** All elements must have a closing tag. For example, `<p>This is a paragraph.</p>`.
- **Case-Sensitivity:** XML tags are case-sensitive (`<Letter>` is different from `<letter>`). Start and end tags must match exactly in case.
- **Proper Nesting:** Elements must be properly nested. For example, `<b><i>bold and italic</i></b>` is correct, while `<b><i>bold and italic</b></i>` is not.
- **Attribute Quotes:** Attribute values must always be enclosed in quotation marks (either single or double).
- **Entity References:** Special characters, like `<` and `&`, must be replaced by predefined entity references (e.g., `&lt;` for `<`) to avoid syntax errors.

Example of an XML Document

xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

The first line is the optional XML declaration. The `<note>` element is the root element, containing four child elements: `<to>`, `<from>`, `<heading>`, and `<body>`.

Common Uses

XML plays a vital role in modern IT systems:

- **Data Transfer:** It is widely used as a standard format for exchanging data between different applications and systems over the Internet, such as in web services (SOAP).
- **Configuration Files:** Many applications, including Android mobile apps, use XML files to store configuration settings and define layouts.
- **Document Formats:** Microsoft Office uses XML-based formats like `.docx`, `.xlsx`, and `.pptx`.
- **Other Standards:** Many other languages and standards are based on XML, including RSS, SVG (Scalable Vector Graphics), and XSLT (XML Transformations).

Related Technologies and Standards

- **DTD and XML Schema (XSD):** These are schema languages used to define the legal structure, elements, and attributes of an XML document, allowing for data validation.
- **XSLT and XPath:** XSLT is a language for transforming XML documents into other formats like HTML, while XPath is used to navigate and select elements within an XML document.
- **Parsers and APIs:** Programmers use APIs like the Document Object Model (DOM) or Simple API for XML (SAX) to read, process, and manipulate XML data within their applications

XML DOCUMENTS STRUCTURES

An XML document is structured as a **hierarchical tree** of elements, starting from a single **root element** and branching to nested child elements. This structure makes it both human-readable and machine-readable for storing and transporting data.
The main components and rules of an XML document structure are:

Core Components

- **Prolog (Optional):** An XML document can optionally start with an XML declaration that specifies the XML version and character encoding, such as `<?xml version="1.0" encoding="UTF-8"?>`. If present, it must be the very first line of the document.

- **Root Element:** Every XML document must have exactly one top-level "root" element that encloses all other elements.
- **Elements:** Elements are the building blocks, defined by start tags (e.g., `<book>`) and end tags (e.g., `</book>`). They can contain text, attributes, or other nested elements.
- **Attributes:** Attributes provide additional information about an element and are included within the start tag as name/value pairs (e.g., `<book category="cooking">`). Attribute values must always be quoted.
- **Hierarchy:** Elements are nested within each other to create parent-child relationships, forming a tree structure. Elements at the same level are called siblings.

Key Syntax Rules

For an XML document to be considered "well-formed" (syntactically correct), it must adhere to specific rules:

- **Proper Nesting:** All elements must be properly nested. The child element must close before its parent element closes (e.g., `<b><i>text</i></b>` is correct, `<b><i>text</b></i>` is not).
- **Matching Tags:** Every start tag must have a matching end tag.
- **Case-Sensitivity:** XML tags are case-sensitive. `<Message>` is different from `<message>`.
- **Quoted Attributes:** Attribute values must always be enclosed in single or double quotation marks.

Example Document Structure

xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<bookstore> <!-- Root Element -->
    <book category="cooking"> <!-- Child element of bookstore, with an attribute -->
        <title lang="en">Everyday Italian</title> <!-- Child element of book -->
        <author>Giada De Laurentiis</author>
        <year>2005</year>
        <price>30.00</price>
    </book>
    <book category="children">
        <title lang="en">Harry Potter</title>
        <author>J.K. Rowling</author>
        <year>2005</year>
        <price>29.99</price>
    </book>
</bookstore>
```

This example, from W3Schools, shows how `<bookstore>` is the root, and it contains multiple `<book>` elements as children, which in turn have their own children like `<title>`, `<author>`, etc.

XML NAME SPACES

XML namespaces are a mechanism used to **avoid name conflicts** between elements and attributes in an XML document, particularly when combining XML from different sources or vocabularies. A namespace is a collection of names identified by a unique Uniform Resource Identifier (URI).

How Namespaces Work

Namespaces are declared using the reserved `xmlns` attribute.

- **Prefixes**: A prefix is used as a shorthand for the long, unique URI within the document. The namespace declaration links the prefix to the URI using the syntax `xmlns:prefix="URI"`. This prefix is then prepended to element and attribute names, separated by a colon, to indicate they belong to that specific namespace (e.g., `<h:table>`).
- **URIs as Identifiers**: The URI acts purely as a unique identifier and does not necessarily need to point to an actual, resolvable web page or schema document, although companies often use a URL for clarity. The URI ensures global uniqueness.
- **Scope**: A namespace declaration is effective for the element where it is declared and all its descendants, unless overridden by another declaration.

Types of Declarations

- **Prefixed Namespace**: Declared with a prefix (`xmlns:prefix="..."`), it applies only to elements and attributes explicitly using that prefix.
- **Default Namespace**: Declared without a prefix (`xmlns="..."`), it applies to all unprefixed elements within its scope. It does not, however, apply to unprefixed attributes, which are considered to be in no namespace.

Example

xml

```xml
<bookstore xmlns:bk="urn:loc.gov:books" xmlns:isbn="urn:ISBN:0-395-36341-6">
  <bk:book>
    <bk:title>Cheaper by the Dozen</bk:title>
    <isbn:number>1568491379</isbn:number>
  </bk:book>
</bookstore>
```

In this example:

- `bk:book` and `bk:title` elements belong to the `"urn:loc.gov:books"` namespace.
- `isbn:number` belongs to the `"urn:ISBN:0-395-36341-6"` namespace.

Namespaces are fundamental to XML technologies like XSLT and XML Schema, as they allow for the combination and validation of different vocabularies in a single document.

XML DOCUMENT TYPE DEFINITION

A **Document Type Definition (DTD)** is a set of rules that defines the legal structure, elements, attributes, and entities of an XML document. It acts as a blueprint, allowing an XML parser to validate that a document conforms to a specific, standardized format.

Purpose and Function

The primary purposes of a DTD are:

- **Structure Validation**: It defines which elements can appear, their order, how they are nested, and how many times they can occur (e.g., once, zero or more times, etc.).
- **Data Integrity and Consistency**: By defining a standard structure, independent groups can agree on a common DTD for data interchange, ensuring that data received from external sources is in a valid and predictable format.
- **Defining Building Blocks**: It declares the fundamental components of the XML document, including:
  - **Elements**: The tags used to structure the data.
  - **Attributes**: The properties associated with elements, including their types (though DTDs only support a limited range, primarily `CDATA` for character data) and default values.
  - **Entities**: Shortcuts for special characters or frequently used text strings (e.g., ` ` for a non-breaking space).

An XML document that adheres to general XML syntax rules is considered "well-formed". If it also conforms to the rules specified in an associated DTD, it is considered a "valid" XML document.

DTD Declaration

A DTD is linked to an XML document using a `DOCTYPE` declaration, which must appear at the beginning of the document, immediately after the XML header. DTDs can be declared in two ways:

- **Internal DTD**: The DTD is embedded directly within the XML file itself, enclosed in square brackets within the `DOCTYPE` declaration.

xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE note [
  <!ELEMENT note (to,from,heading,body)>
  <!ELEMENT to (#PCDATA)>
  <!ELEMENT from (#PCDATA)>
  <!ELEMENT heading (#PCDATA)>
  <!ELEMENT body (#PCDATA)>
]>
<note>
  <!-- ... document content ... -->
</note>
```

- **External DTD**: The DTD is located in a separate file (usually with a `.dtd` extension) and referenced by the XML document using the `SYSTEM` keyword and a file path or URL.

xml

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE note SYSTEM "note.dtd">
<note>
  <!-- ... document content ... -->
</note>
```

Limitations and Alternatives

While DTDs are still in use, particularly in publishing and legacy systems due to their simplicity, they have limitations:

- They use a non-XML syntax, which means they cannot be parsed or manipulated using standard XML tools.
- They offer limited data typing (mostly character data).
- They do not support namespaces.

These limitations led to the development of more powerful alternatives, such as XML Schema Definition (XSD) and RELAX NG, which are written in XML, support complex data types and namespaces, and offer more robust validation capabilities.

XML SCHEMA

An **XML Schema**, often referred to as an **XML Schema Definition (XSD)**, is a document that describes the structure and constraints for a class of XML documents. It defines the "legal" building blocks of an XML file, including elements, attributes, their relationships, and the data types they can contain.

Core Concepts

- **Validation:** An XML document with correct syntax is "well-formed". A document is considered "valid" only if it is well-formed *and* conforms to the rules defined in its associated schema. The process of checking for conformance is called validation.
- **Data Typing:** A key advantage of XSD over older technologies like DTD (Document Type Definition) is its support for a rich set of data types (e.g., `xs:string`, `xs:integer`, `xs:date`, `xs:boolean`). This makes it easier to ensure data correctness and define restrictions on data values.
- **XML Syntax:** XSDs themselves are written in XML syntax, so they do not require learning a separate language and can be edited and manipulated with standard XML tools.

- **Namespaces:** XSD fully supports XML namespaces, which helps avoid name clashes when combining elements from different XML vocabularies.

Key Components

An XML Schema is comprised of several components that define the structure of an XML instance document:

- **Element Declarations:** Define the names of elements that can appear, their types (simple or complex), and occurrence constraints (`minOccurs`, `maxOccurs`).
- **Attribute Declarations:** Define the names and simple types of attributes an element can have, including optional default or fixed values.
- **Simple Type Definitions:** Used for elements or attributes that contain only text and no child elements or attributes (e.g., a price, a date).
- **Complex Type Definitions:** Used for elements that contain child elements and/or attributes, defining their structure and content model.
- **Model Groups:** Used to define the order or grouping of child elements (e.g., `<xs:sequence>`, `<xs:choice>`, `<xs:all>`).
- **Annotations:** Provide human-readable documentation or machine-targeted information within the schema itself.

Why Use an XML Schema?

- **Guaranteed Data Quality:** It enforces rules to ensure the reliability and consistency of data being exchanged between different systems.
- **Interoperability:** It allows independent groups to agree on a common, standardized format for data interchange, ensuring mutual understanding.
- **Code Generation/Data Binding:** Tools can use a schema to automatically generate programming language classes, allowing developers to work with XML data as objects within their applications.
- **Improved Tool Support:** Many XML editors and development environments use the schema to provide features like validation, content assistance (IntelliSense), and graphical editing interfaces.

DISPLAYING XML DOCUMENTS

XML documents can be displayed using **web browsers, text editors, or specialized XML editors**. The display method varies depending on whether the goal is to view the raw data, format it for readability, or present it as a styled webpage.

Viewing Raw XML Data

- **Web Browsers:** Most modern web browsers like Chrome, Firefox, and Edge can open XML files directly. They typically display the data in a color-coded, collapsible tree structure for easy navigation.
  - To view, simply double-click the `.xml` file or drag it into an open browser window.
  - To see the raw source, you can use the "View Page Source" or "View Source" option in the browser menu.
- **Text Editors:** Any basic text editor, such as Notepad or Notepad++, can open and display the raw XML code. These editors are useful for creating or editing the file but do not offer specialized formatting without extensions.
- **XML-aware Editors:** Specialized editors (like oXygen or Visual Studio Code with an XML extension) provide advanced features like syntax highlighting, validation against a schema, and alternate document tree views, which is ideal for development and detailed work.

Styling XML Documents

By default, XML does not carry information about how to display the data, as its tags are not predefined like HTML tags. To present XML data in a visually appealing or structured format, you can link it to a stylesheet:

- **Using CSS (Cascading Style Sheets):** CSS can be used to define style rules for XML elements, such as font, color, and layout (e.g., specifying an element as `display: block`). This is a simpler method for styling the raw data for better readability in a browser.
  - To link a CSS file, add the following processing instruction to your XML document: `<?xml-stylesheet type="text/css" href="stylesheet.css"?>`.
- **Using XSLT (Extensible Stylesheet Language Transformations):** XSLT is a more powerful method used to transform XML data into another format, such as HTML, plain text, or another XML document. This allows for complex presentation logic, including adding/removing elements, sorting data, and making decisions about content display.
  - To link an XSLT file, use the instruction: `<?xml-stylesheet type="text/xsl" href="transform.xsl"?>`.

Displaying XML Data within an HTML Page

To display XML data as part of a larger, styled HTML page, you typically use client-side scripting or server-side programming:

- **JavaScript:** You can use JavaScript (often with AJAX or the Fetch API) to fetch the XML data and then manipulate the HTML DOM to display the information in a specific format, such as an HTML table or `div` element.
- **Server-Side:** Technologies like PHP can read an XML file on the server and generate an HTML output to send to the browser.

WHAT IS XSL

XSL (Extensible Stylesheet Language) is a W3C standard styling language used to define how XML documents are formatted, transformed, or presented. Unlike CSS, which adds style to HTML, XSL is designed specifically to manipulate and transform XML data structures into HTML, text, or other XML formats.

Key Components of XSL:

- **XSLT (XSL Transformations):** A language for transforming XML documents into other formats like HTML or JSON.
- **XPath (XML Path Language):** A language for navigating, selecting, and filtering nodes within an XML document.
- **XSL-FO (XSL Formatting Objects):** A language for formatting XML content for pagination and printing.

Key Features and Uses:

- **Data Transformation:** XSLT can rearrange, add, or delete elements from the original XML structure.
- **Styling XML:** It acts as a style sheet for XML, defining how tags (which have no default display) appear in browsers.
- **Flexibility:** It is more powerful than CSS because it can create a completely different structure from the source XML, such as generating tables from data, sorting data, or applying conditional logic.
- **Components:** While often used interchangeably with XSLT, XSL technically represents the broader specification, with XSLT performing the transformations.

WHAT IS CSS

CSS (**C**ascading **S**tyle **S**heets) is a stylesheet language used to describe the presentation and visual formatting of a document written in a markup language, most commonly HTML. It is a core technology of the World Wide Web, alongside HTML and JavaScript.

Key Functions

CSS is essentially the "stylist" of a website. It separates the content and structure (defined by HTML) from its design, enabling developers to control:

- **Colors** and backgrounds.
- **Fonts**, their size, style, and spacing.
- **Layouts** and the positioning of elements on the page, including advanced tools like Flexbox and Grid for multi-column layouts.
- **Responsiveness**, allowing web pages to adapt their look and feel to different devices and screen sizes, such as mobile phones, tablets, and printers.
- Animations and special visual effects.

The "Cascading" Aspect

The term "cascading" refers to the specific priority scheme the browser uses to resolve conflicts between multiple style rules that might apply to the same element. Styles can come from a browser's default settings, user preferences, or the website author's defined styles (inline, internal, or external), with more specific or later-defined rules taking precedence.

Benefits

- **Separation of Concerns:** Keeps the structure (HTML) separate from the presentation (CSS), making code cleaner, easier to read, and more maintainable.
- **Efficiency:** A single external CSS file can control the look of an entire website, allowing for site-wide design changes by modifying just one file.
- **Improved Performance:** External CSS files can be cached by the browser, reducing load times when users navigate between pages of a website.

Implementation

CSS can be added to HTML documents in three main ways:

1. **Inline CSS:** Using the `style` attribute directly within a single HTML element.
2. **Internal CSS:** Placing the CSS rules within a `<style>` element in the `<head>` section of an HTML document, applying only to that single page.
3. **External CSS:** Linking an external `.css` file to the HTML document, which is the preferred method for styling multiple pages consistently.

XSL AND CSS IN XML

Both XSL and CSS can be used to style XML documents, but they serve different purposes and offer different levels of control.

XSL (Extensible Stylesheet Language)

XSL is a family of standards (XSLT, XPath, XSL-FO) primarily designed for transforming and formatting XML data.

- **XSLT (XSL Transformations):** This is the most widely used part of XSL. XSLT is a language used to transform an XML source document into a new document, which can be HTML, plain text, or a different XML structure. This transformation capability is powerful because it allows you to restructure, add, remove, and sort elements before applying styles.
- **XPath (XML Path Language):** Used by XSLT to navigate and select specific elements or attributes within an XML document.
- **XSL-FO (XSL Formatting Objects):** A language for specifying detailed page layouts and formatting, typically used for generating print-oriented formats like PDFs. Browser support for XSL-FO is limited, and CSS3 is now often considered a replacement for some of its functionality.

CSS (Cascading Style Sheets)

CSS is a style sheet language used to describe the presentation of a document (including HTML and XML).

- **Direct Styling:** CSS can be applied directly to an XML document to define the font, color, and layout of its elements.
- **Limitations:** CSS is primarily for styling and has a reasonably linear structure. It cannot be used to transform or dynamically rearrange the XML document's structure, unlike XSLT. The formatting object tree in CSS is largely the same as the source tree.

How They Work Together

XSL and CSS can be used in combination:

- **Server-side transformation:** The most common approach involves using XSLT on the server to transform the XML data into an HTML document, which is then styled using CSS. This approach ensures compatibility across various user agents (browsers).
- **Client-side styling:** An XML file can link directly to a CSS file using a processing instruction: `<?xml-stylesheet type="text/css" href="style.css"?>`.
- **Integrated approach:** An XSLT stylesheet itself can generate an HTML output that includes a link to a CSS file, allowing the CSS to style the resulting HTML elements.

In summary: use **CSS** for simple styling of XML with a linear structure, or for styling the HTML output of a transformation. Use **XSL** (specifically XSLT) when you need to transform, filter, sort, or reorganize the structure of the XML data before it is displayed.