Unit - 2

Dom: The Document Object Model (DOM) defines the logical structure of documents and the way a document is accessed and manipulated.

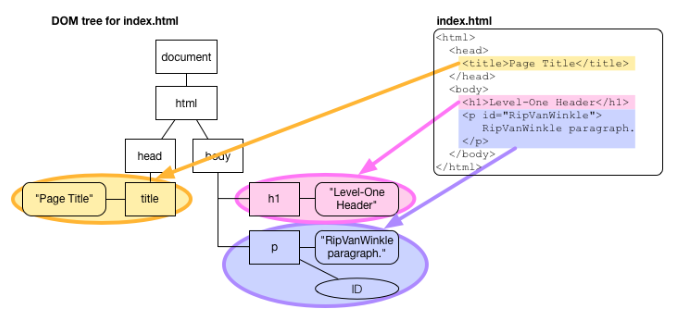
JavaScript interacts with HTML document indirectly by interacting with the **DOM**. With the document object model, JavaScript gets all the power it needs to create dynamic HTML:

* **JavaScript can change all the HTML elements in the page**
* **JavaScript can change all the HTML attributes in the page**
* **JavaScript can change all the CSS styles in the page**
* **JavaScript can remove existing HTML elements and attributes**
* **JavaScript can add new HTML elements and attributes**
* **JavaScript can react to all existing HTML events in the page**
* **JavaScript can create new HTML events in the page**

Link: https://simplesnippets.tech/what-is-document-object-modeldom-how-js-interacts-with-dom/

The **HTML DOM** is an **API** (Programming Interface) for **JavaScript**:

* JavaScript can add/change/remove HTML elements
* JavaScript can add/change/remove HTML attributes
* JavaScript can add/change/remove CSS styles
* JavaScript can react to HTML events
* JavaScript can add/change/remove HTML events

**Link**: https://egghead.io/learn/javascript/the-dom

Link: https://www.w3schools.com/xml/xml\_syntax.asp

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.

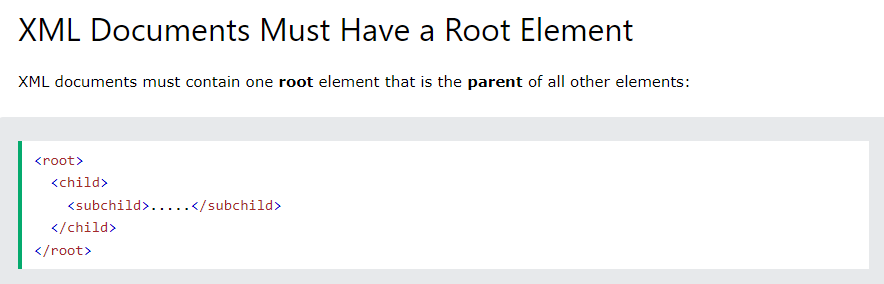
The XML above is quite self-descriptive:

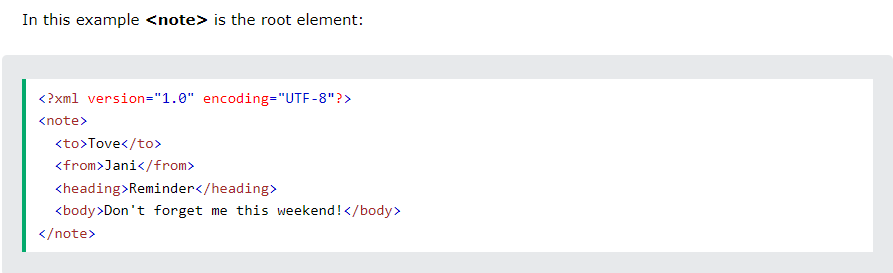
* It has sender information
* It has receiver information
* It has a heading
* It has a message body

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 Syntax Rules:



The XML prolog is optional. If it exists, it must come first in the document.

## All XML Elements Must Have a Closing Tag

In XML, it is illegal to omit the closing tag. All elements **must** have a closing tag:

## XML Tags are Case Sensitive

XML tags are case sensitive. The tag <Letter> is different from the tag <letter>.

Opening and closing tags must be written with the same case:

<message>This is correct</message>

## XML Elements Must be Properly Nested

<b><i>This text is bold and italic</i></b>

In the example above, "Properly nested" simply means that since the <i> element is opened inside the <b> element, it must be closed inside the <b> element.

## Entity References

Some characters have a special meaning in XML.

If you place a character like "<" inside an XML element, it will generate an error because the parser interprets it as the start of a new element.

This will generate an XML error:

<message>salary < 1000</message>

To avoid this error, replace the "<" character with an **entity reference**:

<message>salary &lt; 1000</message>

There are 5 pre-defined entity references in XML:

|  |  |  |
| --- | --- | --- |
| &lt; | < | less than |
| &gt; | > | greater than |
| &amp; | & | ampersand |
| &apos; | ' | apostrophe |
| &quot; | " | quotation mark |

## Comments in XML

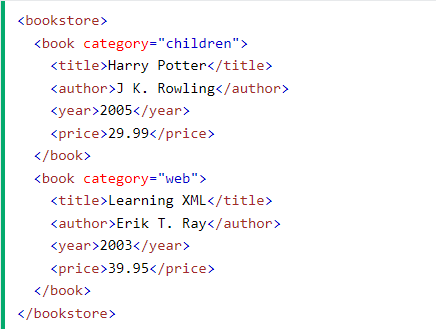
The syntax for writing comments in XML is similar to that of HTML:

<!-- This is a comment -->

## White-space is preserved in XML

XML does not truncate multiple white-spaces (HTML truncates multiple white-spaces to one single white-space):

|  |  |
| --- | --- |
| XML: | Hello           Tove |
| HTML: | Hello Tove |



In the example above:

<title>, <author>, <year>, and <price> have **text content** because they contain text (like 29.99).

<bookstore> and <book> have **element contents**, because they contain elements.

<book> has an **attribute** (category="children").

# **XML Attributes**

## XML Attributes Must be Quoted

Attribute values must always be quoted. Either single or double quotes can be used.

For a person's gender, the <person> element can be written like this:

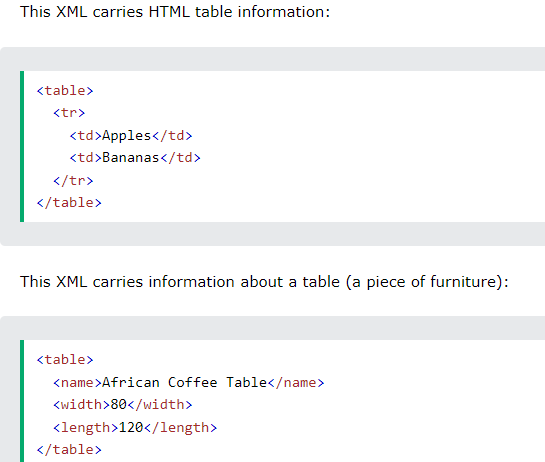
<person gender="female">

or like this:

<person gender='female'>

# **XML Namespaces**

In XML, element names are defined by the developer. This often results in a conflict when trying to mix XML documents from different XML applications.



If these XML fragments were added together, there would be a name conflict. Both contain a <table> element, but the elements have different content and meaning.

## Solving the Name Conflict Using a Prefix

Name conflicts in XML can easily be avoided using a name prefix.

This XML carries information about an HTML table, and a piece of furniture:

## XML Namespaces - The xmlns Attribute

When using prefixes in XML, a **namespace** for the prefix must be defined.

The namespace can be defined by an **xmlns** attribute in the start tag of an element.

The namespace declaration has the following syntax. xmlns:*prefix*="*URI*".



# **Displaying XML**

## Viewing XML Files

<?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>

Look at the XML file above in your browser: [note.xml](https://www.w3schools.com/xml/note.xml)

Most browsers will display an XML document with color-coded elements.

Often a plus (+) or minus sign (-) to the left of the elements can be clicked to expand or collapse the element structure.

To view raw XML source, try to select "View Page Source" or "View Source" from the browser menu.

# **XML HttpRequest**

The XMLHttpRequest Object

The XMLHttpRequest object can be used to request data from a web server.

The XMLHttpRequest object is **a developers dream**, because you can:

* Update a web page without reloading the page
* Request data from a server - after the page has loaded
* Receive data from a server  - after the page has loaded
* Send data to a server - in the background

## Sending an XMLHttpRequest

A common JavaScript syntax for using the XMLHttpRequest object looks much like this:

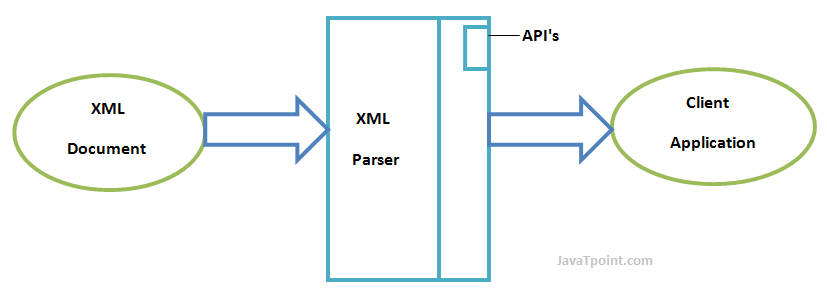
# 

# **XML Parsers**

An XML parser is a software library or package that provides interfaces for client applications to work with an XML document. The XML Parser is designed to read the XML and create a way for programs to use XML.

XML parser validates the document and check that the document is well formatted.

Let's understand the working of XML parser by the figure given below:



## Types of XML Parsers

These are the two main types of XML Parsers:

1. DOM
2. SAX

## DOM (Document Object Model)

A DOM document is an object which contains all the information of an XML document. It is composed like a tree structure. The DOM Parser implements a DOM API. This API is very simple to use.

### **Features of DOM Parser**

A DOM Parser creates an internal structure in memory which is a DOM document object and the client applications get information of the original XML document by invoking methods on this document object.

DOM Parser has a tree based structure.

### **Advantages**

1) It supports both read and write operations and the API is very simple to use.

2) It is preferred when random access to widely separated parts of a document is required.

### **Disadvantages**

1) It is memory inefficient. (consumes more memory because the whole XML document needs to loaded into memory).

2) It is comparatively slower than other parsers.

## SAX (Simple API for XML)

A SAX Parser implements SAX API. This API is an event based API and less intuitive.

### **Features of SAX Parser**

It does not create any internal structure.

Clients does not know what methods to call, they just overrides the methods of the API and place his own code inside method.

It is an event based parser, it works like an event handler in Java.

### **Advantages**

1) It is simple and memory efficient.

2) It is very fast and works for huge documents.

### **Disadvantages**

1) It is event-based so its API is less intuitive.

2) Clients never know the full information because the data is broken into pieces.

## What is XPath?

XPath is a major element in the XSLT standard.

XPath can be used to navigate through elements and attributes in an XML document.

* XPath stands for XML Path Language
* XPath uses "path like" syntax to identify and navigate nodes in an XML document
* XPath contains over 200 built-in functions
* XPath is a major element in the XSLT standard
* XPath is a W3C recommendation

XPath expressions can also be used in JavaScript, Java, XML Schema, PHP, Python, C and C++, and lots of other languages.

XPath (XML Path Language) is a query language for selecting nodes from an XML document. Here's a simple example to demonstrate how XPath can be used to navigate and select parts of an XML document.

# **XSLT Introduction**

XSL (eXtensible Stylesheet Language) is a styling language for XML.

XSLT stands for XSL Transformations.

**XSLT** is a language for transforming XML documents.

XSL stands for E**X**tensible **S**tylesheet **L**anguage.

The World Wide Web Consortium (W3C) started to develop XSL because there was a need for an XML-based Stylesheet Language.

## CSS = Style Sheets for HTML

HTML uses predefined tags. The meaning of, and how to display each tag is well understood.

CSS is used to add styles to HTML elements.

## XSL = Style Sheets for XML

XML does not use predefined tags, and therefore the meaning of each tag is not well understood.

A <table> element could indicate an HTML table, a piece of furniture, or something else - and browsers do not know how to display it!

So, XSL describes how the XML elements should be displayed.

## How Does it Work?

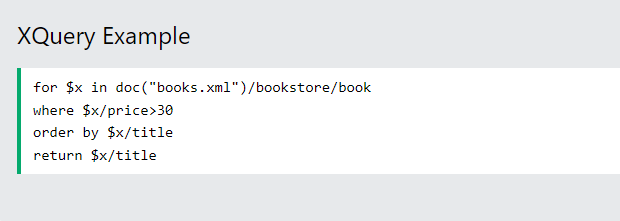
In the transformation process, XSLT uses XPath to define parts of the source document that should match one or more predefined templates. When a match is found, XSLT will transform the matching part of the source document into the result document.

## What is XQuery?

## XQuery is a powerful language designed to query and manipulate XML data. It can be used to retrieve, transform, and manipulate XML documents. Here's a simple example of how XQuery can be used.

XQuery is to XML what SQL is to databases.

XQuery is designed to query XML data.



To run this XQuery, you'll need an XQuery processor. Tools like BaseX, Saxon, or eXist-db can be used to execute XQuery expressions.

What is XQuery?

|  |  |
| --- | --- |
| XQuery | * XQuery is ***the*** language for querying XML data * XQuery for XML is like SQL for databases * XQuery is built on XPath expressions * XQuery is supported by all major databases * XQuery is a W3C Recommendation |

XQuery can be used to:

* Extract information to use in a Web Service
* Generate summary reports
* Transform XML data to XHTML
* Search Web documents for relevant information

# **XML, XLink**

XLink is used to create hyperlinks in XML documents.

|  |  |
| --- | --- |
| XPath | * XLink is used to create hyperlinks within XML documents * Any element in an XML document can behave as a link * With XLink, the links can be defined outside the linked files * XLink is a W3C Recommendation |

