XML

**Introduction**:

* Extensible Markup Language
* Designed to store and transport data
* Stores data in plain text format 🡪 hardware & software independent
* Designed to be both human & machine readable (self-descriptive & easy to understand)
* Doesn’t actually do anything – just information wrapped in tags

XML vs. HTML:

* XML – designed to carry data – focusses on what data is
* Uses author defined tags
* HTML – designed to display data – focusses on how the data looks
* Uses predefined tags

**How Used**:

* Doesn’t carry any info about how to be displayed
* The same XML data can be used in many presentation scenarios
* Full separation between data & presentation
* Therefore 🡪 is often used to separate data from presentation
* To complement HTML
* XML used to store & transport data, HTML used to format & display data

**XML Tree**:

XML documents form a tree structure that starts at "the root" and branches to "the leaves"

* XML Documents are formed as element trees
* XML tree starts at a Root Element & branches from the root to Child Elements
* All elements can have Sub Elements
* All elements can have text content and attributes

Self-Describing Syntax:

* Prolog – defines the XML version and the character encoding

<?xml version="1.0" encoding="UTF-8**"**?>

* The next line is the root element – the tag that enwraps the entire document

<bookstore>  
  <book category="cooking">\*sub elements\*</book>  
  <book category="children">\*sub elements\*</book>  
</bookstore>

**Syntax**: (Defines “Well-Formed” XML Documents)

* XML Documents must contain 1 root element that is the parent of all other elements
* XML Prolog – option BUT must be first if included
* XML Tags are case sensitive
* XML Elements must have a closing tag (except for the prolog)
* XML Elements must be properly nested (vs. HTML)
* XML Attribute Values must be quoted
* Comments 🡪 <!-- This is a comment --> <!-- This is an invalid -- comment -->
* White Space 🡪 multiple white spaces are not truncated (vs. HTML)
* 5 predefined Entity References

**Elements**:

* An XML element is everything from (& including) the element's start tag, to (& including) the element's end tag
* Can contain Text Content, Attributes, other Elements, or a mix of these

**Attributes**:

* Designed to contain data related to a specific element
* No rules about when to use Elements vs. Attributes
* Reasons to use Elements over Attributes:
* attributes cannot contain multiple values (elements can)
* attributes cannot contain tree structures (elements can)
* attributes are not easily expandable (for future changes)
* Should use Attributes for Metadata
* metadata should be stored as attributes
* the data itself should be stored as elements

**XLM Namespaces**:

* Provide a method to avoid element name conflicts
* Prefixes:
* used to avoid name conflicts
* a namespace for the prefix must be defined
* can be defined by an xmlns attribute in the start tag of an element
* xmlns Attribute:
* syntax: xmlns:*prefix*="*URI*"
* When a namespace is defined for an element, all child elements with the same prefix are associated with the same namespace

**URI**: ( **Uniform Resource Identifier** )

* A string of characters which identifies an Internet Resource

\*Most common URI – **Uniform Resource Locator** (URL) - identifies an Internet domain address

**Display**:

* XML doesn’t contain any info about how to display the data
* Browsers display an XML Document as it
* Invalid XML File (one w/ an error):
* Some browsers will report the error, some will display it, or display it incorrectly
* Formatting XML w/ CSS is not recommended – USE JAVASCRIPT OR XSLT INSTEAD

**XML HttpRequest**:

XMLHttpRequest Object:

* Can be used to request data from a web server
* Allows you to:
* 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

**XML Parser**:

* All major browsers have a built-in XML Parser to access & manipulate XML
* The **XML DOM** – defines properties & methods for accessing & editing XML

\*XML DOM Tutorial: \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

What is the DOM?

* Document Object Model – is a platform & language-neutral interface that allows programs & scripts to dynamically access & update the content, structure, & style of a document

XML DOM:

* A standard object model for XML
* A standard programming interface for XML
* Platform- and language-independent
* XML Elements can be accessed through the XML DOM
* Defines a standard for accessing & manipulating XML Documents
* How to get, change, add, or delete XML Elements

Programming Interface:

* The DOM models XML as a set of node objects
* These nodes can then be accessed w/ JS
* The programming interface to the DOM is defined by a set standard properties and methods

**DOM Nodes**:

* According to the XML DOM – everything in an XML document is a node
* the entire document is a document node
* every XML element is an element node
* the text in the XML elements are text nodes
* every attribute is an attribute node
* comments are comment nodes
* Text is always stored in text nodes
* **<year>2005</year>** - the element node <year> holds a text node with the value "2005"
* "2005" is **not** the value of the <year> element!

XML DOM Node Tree:

* XML DOM views an XML document as a tree-structure called a Node-Tree
* All nodes can be accessed through the tree & their contents can be modified or deleted, and new elements can be created

**Accessing Nodes**: