[*XML Tutorial* 16](#_Toc483672460)

[*XML Example 1* 16](#_Toc483672461)

[*Why Study XML?* 16](#_Toc483672462)

[*What You Will Learn* 16](#_Toc483672463)

[*Important XML Standards* 16](#_Toc483672464)

[*Learn by Examples* 17](#_Toc483672465)

[*XML Quiz Test* 17](#_Toc483672466)

[*Introduction to XML* 17](#_Toc483672467)

[*What is XML?* 17](#_Toc483672468)

[*XML Does Not DO Anything* 17](#_Toc483672469)

[*The Difference Between XML and HTML* 18](#_Toc483672470)

[*XML Does Not Use Predefined Tags* 18](#_Toc483672471)

[*XML is Extensible* 18](#_Toc483672472)

[*XML Simplifies Things* 18](#_Toc483672473)

[*XML is a W3C Recommendation* 19](#_Toc483672474)

[*How Can XML be Used?* 19](#_Toc483672475)

[*XML Separates Data from Presentation* 19](#_Toc483672476)

[*XML is Often a Complement to HTML* 19](#_Toc483672477)

[*XML Separates Data from HTML* 19](#_Toc483672478)

[*Transaction Data* 19](#_Toc483672479)

[*XML Tree* 20](#_Toc483672480)

[*XML Tree Structure* 20](#_Toc483672481)

[*Self-Describing Syntax* 21](#_Toc483672482)

[*XML Syntax Rules* 21](#_Toc483672483)

[*XML Documents Must Have a Root Element* 21](#_Toc483672484)

[*The XML Prolog* 22](#_Toc483672485)

[*All XML Elements Must Have a Closing Tag* 22](#_Toc483672486)

[*XML Tags are Case Sensitive* 22](#_Toc483672487)

[*XML Elements Must be Properly Nested* 22](#_Toc483672488)

[*XML Attribute Values Must be Quoted* 23](#_Toc483672489)

[*Entity References* 23](#_Toc483672490)

[*Comments in XML* 24](#_Toc483672491)

[*White-space is Preserved in XML* 24](#_Toc483672492)

[*XML Stores New Line as LF* 24](#_Toc483672493)

[*Well Formed XML* 24](#_Toc483672494)

[*XML Elements* 24](#_Toc483672495)

[*What is an XML Element?* 24](#_Toc483672496)

[*Empty XML Elements* 25](#_Toc483672497)

[*XML Naming Rules* 26](#_Toc483672498)

[*Best Naming Practices* 26](#_Toc483672499)

[*Naming Styles* 26](#_Toc483672500)

[*XML Elements are Extensible* 27](#_Toc483672501)

[*XML Attributes* 27](#_Toc483672502)

[*XML Attributes Must be Quoted* 27](#_Toc483672503)

[*XML Elements vs. Attributes* 28](#_Toc483672504)

[*My Favorite Way* 28](#_Toc483672505)

[*Avoid XML Attributes?* 29](#_Toc483672506)

[*Don't end up like this:* 29](#_Toc483672507)

[*XML Attributes for Metadata* 29](#_Toc483672508)

[*XML Namespaces* 29](#_Toc483672509)

[*Name Conflicts* 30](#_Toc483672510)

[*Solving the Name Conflict Using a Prefix* 30](#_Toc483672511)

[*XML Namespaces - The xmlns Attribute* 30](#_Toc483672512)

[*Uniform Resource Identifier (URI)* 32](#_Toc483672513)

[*Default Namespaces* 32](#_Toc483672514)

[*Namespaces in Real Use* 32](#_Toc483672515)

[*Displaying XML* 32](#_Toc483672516)

[*Viewing XML Files* 33](#_Toc483672517)

[*Viewing an Invalid XML File* 33](#_Toc483672518)

[*Other XML Examples* 33](#_Toc483672519)

[*Why Does XML Display Like This?* 33](#_Toc483672520)

[*Displaying XML Files with CSS?* 34](#_Toc483672521)

[*XML HttpRequest* 34](#_Toc483672522)

[*The XMLHttpRequest Object* 34](#_Toc483672523)

[*XMLHttpRequest Example* 34](#_Toc483672524)

[*Sending an XMLHttpRequest* 34](#_Toc483672525)

[*Example Explained* 35](#_Toc483672526)

[*XML Parser* 35](#_Toc483672527)

[*XML Parser* 35](#_Toc483672528)

[*Parsing a Text String* 35](#_Toc483672529)

[*The XMLHttpRequest Object* 37](#_Toc483672530)

[*XML DOM* 37](#_Toc483672531)

[*What is the DOM?* 37](#_Toc483672532)

[*The HTML DOM* 38](#_Toc483672533)

[*The XML DOM* 38](#_Toc483672534)

[*XML and XPath* 39](#_Toc483672535)

[*What is XPath?* 39](#_Toc483672536)

[*XPath Path Expressions* 40](#_Toc483672537)

[*XPath is Used in XSLT* 40](#_Toc483672538)

[*XPath Example* 40](#_Toc483672539)

[*XML and XSLT* 42](#_Toc483672540)

[*Displaying XML with XSLT* 42](#_Toc483672541)

[*XSLT Example* 42](#_Toc483672542)

[*XML and XQuery* 42](#_Toc483672543)

[*What is XQuery?* 42](#_Toc483672544)

[*What is XQuery?* 42](#_Toc483672545)

[*XQuery is About Querying XML* 43](#_Toc483672546)

[*XQuery and XPath* 43](#_Toc483672547)

[*XQuery - Examples of Use* 43](#_Toc483672548)

[*XQuery is a W3C Recommendation* 43](#_Toc483672549)

[*XML, XLink and XPointer* 43](#_Toc483672550)

[*XLink Browser Support* 44](#_Toc483672551)

[*XLink Syntax* 44](#_Toc483672552)

[*XLink Example* 44](#_Toc483672553)

[*XLink - Going Further* 45](#_Toc483672554)

[*XLink Attribute Reference* 45](#_Toc483672555)

[*XPointer* 46](#_Toc483672556)

[*XPointer Browser Support* 46](#_Toc483672557)

[*XPointer Example* 46](#_Toc483672558)

[*XML Validator* 48](#_Toc483672559)

[*Well Formed XML Documents* 48](#_Toc483672560)

[*XML Errors Will Stop You* 48](#_Toc483672561)

[*Syntax-Check Your XML* 48](#_Toc483672562)

[*Valid XML Documents* 48](#_Toc483672563)

[*XML DTD* 49](#_Toc483672564)

[*Valid XML Documents* 49](#_Toc483672565)

[*XML DTD* 49](#_Toc483672566)

[*Using DTD for Entity Declaration* 50](#_Toc483672567)

[*When to Use a DTD/Schema?* 50](#_Toc483672568)

[*When to NOT to Use a DTD/Schema?* 50](#_Toc483672569)

[*XML Schema* 51](#_Toc483672570)

[*XML Schema* 51](#_Toc483672571)

[*XML Schemas are More Powerful than DTD* 51](#_Toc483672572)

[*Why Use an XML Schema?* 52](#_Toc483672573)

[*XML Schemas Support Data Types* 52](#_Toc483672574)

[*XML Schemas use XML Syntax* 52](#_Toc483672575)

[*XML on the Server* 52](#_Toc483672576)

[*Storing XML Files on the Server* 52](#_Toc483672577)

[*Generating XML with PHP* 53](#_Toc483672578)

[*Generating XML with ASP* 53](#_Toc483672579)

[*Generating XML From a Database* 53](#_Toc483672580)

[*Transforming XML with XSLT on the Server* 54](#_Toc483672581)

[*XML Examples* 55](#_Toc483672582)

[*XML Quiz* 55](#_Toc483672583)

[*The Test* 55](#_Toc483672584)

[*Count Your Score* 55](#_Toc483672585)

[*W3Schools XML Certificate* 56](#_Toc483672586)

[*How Does It Work?* 56](#_Toc483672587)

[AJAX 56](#_Toc483672588)

[*AJAX Introduction* 56](#_Toc483672589)

[*AJAX Example* 56](#_Toc483672590)

[*What is AJAX?* 57](#_Toc483672591)

[*How AJAX Works* 57](#_Toc483672592)

[*AJAX - The XMLHttpRequest Object* 58](#_Toc483672593)

[*The XMLHttpRequest Object* 58](#_Toc483672594)

[*Create an XMLHttpRequest Object* 58](#_Toc483672595)

[*Access Across Domains* 58](#_Toc483672596)

[*XMLHttpRequest Object Methods* 59](#_Toc483672597)

[*XMLHttpRequest Object Properties* 59](#_Toc483672598)

[*AJAX - Send a Request To a Server* 60](#_Toc483672599)

[*Send a Request To a Server* 60](#_Toc483672600)

[*GET or POST?* 60](#_Toc483672601)

[*GET Requests* 60](#_Toc483672602)

[*POST Requests* 61](#_Toc483672603)

[*The url - A File On a Server* 61](#_Toc483672604)

[*Asynchronous - True or False?* 61](#_Toc483672605)

[*AJAX - Server Response* 63](#_Toc483672606)

[*The onreadystatechange Property* 63](#_Toc483672607)

[*Using a Callback Function* 64](#_Toc483672608)

[*Server Response Properties* 64](#_Toc483672609)

[*Server Response Methods* 64](#_Toc483672610)

[*The responseText Property* 65](#_Toc483672611)

[*The responseXML Property* 65](#_Toc483672612)

[*The getAllResponseHeaders() Method* 65](#_Toc483672613)

[*The getResponseHeader() Method* 66](#_Toc483672614)

[*AJAX XML Example* 66](#_Toc483672615)

[*AJAX XML Example* 66](#_Toc483672616)

[*Example Explained* 66](#_Toc483672617)

[*AJAX PHP Example* 66](#_Toc483672618)

[*AJAX PHP Example* 67](#_Toc483672619)

[*Example Explained* 67](#_Toc483672620)

[The PHP File - "gethint.php" 67](#_Toc483672621)

[*AJAX ASP Example* 69](#_Toc483672622)

[*AJAX ASP Example* 69](#_Toc483672623)

[*Example Explained* 69](#_Toc483672624)

[The ASP File - "gethint.asp" 69](#_Toc483672625)

[*AJAX Database Example* 71](#_Toc483672626)

[*The AJAX Server Page* 71](#_Toc483672627)

[*XML Applications* 71](#_Toc483672628)

[*The XML Document Used* 72](#_Toc483672629)

[*Display XML Data in an HTML Table* 72](#_Toc483672630)

[*Display the First CD in an HTML div Element* 72](#_Toc483672631)

[DOM 72](#_Toc483672632)

[*XML DOM Tutorial* 72](#_Toc483672633)

[*XML DOM* 72](#_Toc483672634)

[*What is the DOM?* 73](#_Toc483672635)

[*The HTML DOM* 73](#_Toc483672636)

[*The XML DOM* 74](#_Toc483672637)

[*Get the Value of an XML Element* 74](#_Toc483672638)

[*Loading an XML File* 74](#_Toc483672639)

[*Loading an XML String* 74](#_Toc483672640)

[*Programming Interface* 75](#_Toc483672641)

[*XML DOM Properties* 75](#_Toc483672642)

[*XML DOM Methods* 75](#_Toc483672643)

[*XML DOM Nodes* 75](#_Toc483672644)

[*DOM Example* 76](#_Toc483672645)

[*Text is Always Stored in Text Nodes* 76](#_Toc483672646)

[*The XML DOM Node Tree* 76](#_Toc483672647)

[*Node Parents, Children, and Siblings* 77](#_Toc483672648)

[*XML DOM - Accessing Nodes* 78](#_Toc483672649)

[*Accessing Nodes* 79](#_Toc483672650)

[*The getElementsByTagName() Method* 79](#_Toc483672651)

[*DOM Node List* 79](#_Toc483672652)

[*Node Types* 80](#_Toc483672653)

[*Traversing Nodes* 80](#_Toc483672654)

[*Navigating Node Relationships* 80](#_Toc483672655)

[*XML DOM Node Information* 81](#_Toc483672656)

[*Node Properties* 81](#_Toc483672657)

[*The nodeName Property* 81](#_Toc483672658)

[*The nodeValue Property* 82](#_Toc483672659)

[*Get the Value of an Element* 82](#_Toc483672660)

[*The nodeType Property* 82](#_Toc483672661)

[*XML DOM Node List* 82](#_Toc483672662)

[*DOM Node List* 83](#_Toc483672663)

[*Node List Length* 83](#_Toc483672664)

[*DOM Attribute List (Named Node Map)* 84](#_Toc483672665)

[*XML DOM Traverse Node Tree* 84](#_Toc483672666)

[*Traversing the Node Tree* 84](#_Toc483672667)

[*Browser Differences in DOM Parsing* 84](#_Toc483672668)

[*DOM - White Spaces and New Lines* 85](#_Toc483672669)

[*PCDATA - Parsed Character Data* 85](#_Toc483672670)

[*CDATA - (Unparsed) Character Data* 85](#_Toc483672671)

[*XML DOM - Navigating Nodes* 86](#_Toc483672672)

[*Navigating DOM Nodes* 86](#_Toc483672673)

[*DOM - Parent Node* 87](#_Toc483672674)

[*Avoid Empty Text Nodes* 87](#_Toc483672675)

[*XML DOM Get Node Values* 88](#_Toc483672676)

[*Get the Value of an Element* 88](#_Toc483672677)

[*The getElementsByTagName Method* 88](#_Toc483672678)

[*The ChildNodes Property* 89](#_Toc483672679)

[*The nodeValue Property* 89](#_Toc483672680)

[*Get the Value of an Attribute* 89](#_Toc483672681)

[*Get an Attribute Value - getAttribute()* 89](#_Toc483672682)

[*Get an Attribute Value - getAttributeNode()* 90](#_Toc483672683)

[*XML DOM Change Node Values* 90](#_Toc483672684)

[*Change the Value of an Element* 90](#_Toc483672685)

[*Change the Value of a Text Node* 90](#_Toc483672686)

[*Change the Value of an Attribute* 90](#_Toc483672687)

[*Change an Attribute Using setAttribute()* 90](#_Toc483672688)

[*Change an Attribute Using nodeValue* 91](#_Toc483672689)

[*XML DOM Remove Nodes* 91](#_Toc483672690)

[*Remove an Element Node* 91](#_Toc483672691)

[*Remove Myself - Remove the Current Node* 91](#_Toc483672692)

[*Remove a Text Node* 92](#_Toc483672693)

[*Clear a Text Node (nodeValue)* 92](#_Toc483672694)

[*Remove an Attribute Node by Name* 92](#_Toc483672695)

[*Remove Attribute Nodes by Object* 93](#_Toc483672696)

[*XML DOM Replace Nodes* 93](#_Toc483672697)

[*Replace an Element Node* 93](#_Toc483672698)

[*Replace Data In a Text Node* 94](#_Toc483672699)

[*Use the nodeValue Property Instead* 94](#_Toc483672700)

[*XML DOM Create Nodes* 94](#_Toc483672701)

[*Create a New Element Node* 94](#_Toc483672702)

[*Create a New Attribute Node* 95](#_Toc483672703)

[*Create an Attribute Using setAttribute()* 95](#_Toc483672704)

[*Create a Text Node* 95](#_Toc483672705)

[*Create a CDATA Section Node* 96](#_Toc483672706)

[*Create a Comment Node* 96](#_Toc483672707)

[*XML DOM Add Nodes* 97](#_Toc483672708)

[*Add a Node - appendChild()* 97](#_Toc483672709)

[*Insert a Node - insertBefore()* 97](#_Toc483672710)

[*Add a New Attribute* 97](#_Toc483672711)

[*Add Text to a Text Node - insertData()* 98](#_Toc483672712)

[*XML DOM Clone Nodes* 98](#_Toc483672713)

[*Copy a Node* 98](#_Toc483672714)

[*тут пералічаны ўсе прыклады* 99](#_Toc483672715)

[*Дакументацыя па DOM* 99](#_Toc483672716)

[*дакументацыя па xslt* 99](#_Toc483672717)

[*XPath* 99](#_Toc483672718)

[*XPath Tutorial* 99](#_Toc483672719)

[*What is XPath?* 100](#_Toc483672720)

[*XPath Path Expressions* 100](#_Toc483672721)

[*XPath Standard Functions* 100](#_Toc483672722)

[*XPath is Used in XSLT* 100](#_Toc483672723)

[*XPath is a W3C Recommendation* 101](#_Toc483672724)

[*XPath Nodes* 101](#_Toc483672725)

[*XPath Terminology* 101](#_Toc483672726)

[*Nodes* 101](#_Toc483672727)

[*Atomic values* 101](#_Toc483672728)

[*Items* 102](#_Toc483672729)

[*Relationship of Nodes* 102](#_Toc483672730)

[*XPath Syntax* 102](#_Toc483672731)

[*The XML Example Document* 102](#_Toc483672732)

[*Selecting Nodes* 103](#_Toc483672733)

[*Predicates* 104](#_Toc483672734)

[*Selecting Unknown Nodes* 105](#_Toc483672735)

[*Selecting Several Paths* 105](#_Toc483672736)

[*XPath Axes* 106](#_Toc483672737)

[*The XML Example Document* 106](#_Toc483672738)

[*XPath Axes* 106](#_Toc483672739)

[*Location Path Expression* 107](#_Toc483672740)

[*XPath Operators* 108](#_Toc483672741)

[*XPath Operators* 108](#_Toc483672742)

[*XPath Examples* 109](#_Toc483672743)

[*The XML Example Document* 109](#_Toc483672744)

[*Loading the XML Document* 110](#_Toc483672745)

[*Selecting Nodes* 110](#_Toc483672746)

[*Select all the titles* 110](#_Toc483672747)

[*Select the title of the first book* 110](#_Toc483672748)

[*Select all the prices* 110](#_Toc483672749)

[*Select price nodes with price>35* 111](#_Toc483672750)

[*Select title nodes with price>35* 111](#_Toc483672751)

[*XSLT* 111](#_Toc483672752)

[*XSLT Introduction* 111](#_Toc483672753)

[*Online XSLT Editor* 111](#_Toc483672754)

[*What You Should Already Know* 112](#_Toc483672755)

[*XSLT References* 112](#_Toc483672756)

[*XSL(T) Languages* 112](#_Toc483672757)

[*XSL - More Than a Style Sheet Language* 113](#_Toc483672758)

[*What is XSLT?* 113](#_Toc483672759)

[*XSLT Uses XPath* 114](#_Toc483672760)

[*How Does it Work?* 114](#_Toc483672761)

[*XSLT Browser Support* 114](#_Toc483672762)

[*XSLT - Transformation* 114](#_Toc483672763)

[*Correct Style Sheet Declaration* 114](#_Toc483672764)

[*Start with a Raw XML Document* 115](#_Toc483672765)

[*Create an XSL Style Sheet* 115](#_Toc483672766)

[*Link the XSL Style Sheet to the XML Document* 116](#_Toc483672767)

[*XSLT <xsl:template> Element* 116](#_Toc483672768)

[*The <xsl:template> Element* 116](#_Toc483672769)

[*Example* 117](#_Toc483672770)

[*XSLT <xsl:value-of> Element* 117](#_Toc483672771)

[*The <xsl:value-of> Element* 118](#_Toc483672772)

[*XSLT <xsl:for-each> Element* 118](#_Toc483672773)

[*The <xsl:for-each> Element* 118](#_Toc483672774)

[*Filtering the Output* 119](#_Toc483672775)

[*XSLT <xsl:sort> Element* 120](#_Toc483672776)

[*Where to put the Sort Information* 120](#_Toc483672777)

[*Example* 120](#_Toc483672778)

[*XSLT <xsl:if> Element* 121](#_Toc483672779)

[*The <xsl:if> Element* 121](#_Toc483672780)

[*Syntax* 121](#_Toc483672781)

[*Where to Put the <xsl:if> Element* 121](#_Toc483672782)

[*Example* 121](#_Toc483672783)

[*XSLT <xsl:choose> Element* 122](#_Toc483672784)

[*Syntax* 122](#_Toc483672785)

[*Where to put the Choose Condition* 122](#_Toc483672786)

[*Example* 122](#_Toc483672787)

[*XSLT <xsl:apply-templates> Element* 124](#_Toc483672788)

[*Example* 124](#_Toc483672789)

[*XSLT - On the Client* 125](#_Toc483672790)

[*A JavaScript Solution* 125](#_Toc483672791)

[*The XML File and the XSL File* 126](#_Toc483672792)

[*Transforming XML to XHTML in the Browser* 127](#_Toc483672793)

[*XSLT - On the Server* 128](#_Toc483672794)

[*A Cross Browser Solution* 128](#_Toc483672795)

[*The XML File and the XSLT File* 129](#_Toc483672796)

[*PHP Code: Transform XML to XHTML on the Server* 130](#_Toc483672797)

[*ASP Code: Transform XML to XHTML on the Server* 130](#_Toc483672798)

[*XSLT - Editing XML* 131](#_Toc483672799)

[*Open, Edit and Save XML* 131](#_Toc483672800)

[*The XML File and the XSL File* 131](#_Toc483672801)

[*The PHP File* 133](#_Toc483672802)

[*The ASP File* 134](#_Toc483672803)

[*XSLT Examples* 135](#_Toc483672804)

[*XQuery* 135](#_Toc483672805)

[*XQuery Tutorial* 135](#_Toc483672806)

[*What is XQuery?* 136](#_Toc483672807)

[*XQuery Example* 136](#_Toc483672808)

[*What is XQuery?* 136](#_Toc483672809)

[*XQuery is About Querying XML* 136](#_Toc483672810)

[*XQuery and XPath* 136](#_Toc483672811)

[*XQuery can be used to:* 136](#_Toc483672812)

[*XQuery is a W3C Recommendation* 136](#_Toc483672813)

[*XQuery Example* 137](#_Toc483672814)

[*The XML Example Document* 137](#_Toc483672815)

[*How to Select Nodes From "books.xml"?* 137](#_Toc483672816)

[*Functions* 137](#_Toc483672817)

[*Path Expressions* 137](#_Toc483672818)

[*Predicates* 138](#_Toc483672819)

[*XQuery FLWOR Expressions* 138](#_Toc483672820)

[*The XML Example Document* 138](#_Toc483672821)

[*How to Select Nodes From "books.xml" With FLWOR* 138](#_Toc483672822)

[*XQuery FLWOR + HTML* 139](#_Toc483672823)

[*The XML Example Document* 139](#_Toc483672824)

[*Present the Result In an HTML List* 139](#_Toc483672825)

[*XQuery Terms* 141](#_Toc483672826)

[*XQuery Terminology* 141](#_Toc483672827)

[*Nodes* 141](#_Toc483672828)

[*Atomic values* 141](#_Toc483672829)

[*Items* 141](#_Toc483672830)

[*Relationship of Nodes* 141](#_Toc483672831)

[*XQuery Syntax* 143](#_Toc483672832)

[*XQuery Basic Syntax Rules* 143](#_Toc483672833)

[*The XML Example Document* 143](#_Toc483672834)

[*XQuery Conditional Expressions* 143](#_Toc483672835)

[*XQuery Comparisons* 144](#_Toc483672836)

[*XQuery Adding Elements and Attributes* 144](#_Toc483672837)

[*The XML Example Document* 144](#_Toc483672838)

[*Adding Elements and Attributes to the Result* 144](#_Toc483672839)

[*Add HTML Elements and Text* 145](#_Toc483672840)

[*Add Attributes to HTML Elements* 146](#_Toc483672841)

[*XQuery Selecting and Filtering* 146](#_Toc483672842)

[*The XML Example Document* 146](#_Toc483672843)

[*Selecting and Filtering Elements* 147](#_Toc483672844)

[*The for Clause* 147](#_Toc483672845)

[*The let Clause* 148](#_Toc483672846)

[*The where Clause* 148](#_Toc483672847)

[*The order by Clause* 148](#_Toc483672848)

[*The return Clause* 148](#_Toc483672849)

[*XQuery Functions* 149](#_Toc483672850)

[*XQuery Data Types* 149](#_Toc483672851)

[*Examples of Function Calls* 149](#_Toc483672852)

[*XQuery User-Defined Functions* 150](#_Toc483672853)

[*Syntax* 150](#_Toc483672854)

[*DTD* 150](#_Toc483672855)

[*DTD Tutorial* 150](#_Toc483672856)

[*What is a DTD?* 151](#_Toc483672857)

[*Why Use a DTD?* 151](#_Toc483672858)

[*An Internal DTD Declaration* 151](#_Toc483672859)

[*An External DTD Declaration* 151](#_Toc483672860)

[*DTD - XML Building Blocks* 152](#_Toc483672861)

[*Elements* 152](#_Toc483672862)

[*Attributes* 152](#_Toc483672863)

[*Entities* 153](#_Toc483672864)

[*PCDATA* 153](#_Toc483672865)

[*CDATA* 153](#_Toc483672866)

[*DTD - Elements* 154](#_Toc483672867)

[*Declaring Elements* 154](#_Toc483672868)

[*Empty Elements* 154](#_Toc483672869)

[*Elements with Parsed Character Data* 154](#_Toc483672870)

[*Elements with any Contents* 154](#_Toc483672871)

[*Elements with Children (sequences)* 155](#_Toc483672872)

[*Declaring Only One Occurrence of an Element* 155](#_Toc483672873)

[*Declaring Minimum One Occurrence of an Element* 155](#_Toc483672874)

[*Declaring Zero or More Occurrences of an Element* 155](#_Toc483672875)

[*Declaring Zero or One Occurrences of an Element* 156](#_Toc483672876)

[*Declaring either/or Content* 156](#_Toc483672877)

[*Declaring Mixed Content* 156](#_Toc483672878)

[*DTD - Attributes* 156](#_Toc483672879)

[*Declaring Attributes* 156](#_Toc483672880)

[*A Default Attribute Value* 157](#_Toc483672881)

[*#REQUIRED* 157](#_Toc483672882)

[*#IMPLIED* 158](#_Toc483672883)

[*#FIXED* 158](#_Toc483672884)

[*Enumerated Attribute Values* 159](#_Toc483672885)

[*XML Elements vs. Attributes* 159](#_Toc483672886)

[*Use of Elements vs. Attributes* 159](#_Toc483672887)

[*My Favorite Way* 160](#_Toc483672888)

[*Avoid using attributes?* 160](#_Toc483672889)

[*An Exception to my Attribute Rule* 161](#_Toc483672890)

[*DTD - Entities* 162](#_Toc483672891)

[*An Internal Entity Declaration* 162](#_Toc483672892)

[*An External Entity Declaration* 162](#_Toc483672893)

[XML Schema 163](#_Toc483672894)

[*XML Schema Tutorial* 163](#_Toc483672895)

[*What is an XML Schema?* 163](#_Toc483672896)

[*XSD Example* 163](#_Toc483672897)

[*Why Learn XML Schema?* 163](#_Toc483672898)

[*XML Schemas Support Data Types* 164](#_Toc483672899)

[*XML Schemas use XML Syntax* 164](#_Toc483672900)

[*XML Schemas are extensible, because they are written in XML.* 164](#_Toc483672901)

[*XML Schemas Secure Data Communication* 164](#_Toc483672902)

[*Well-Formed is Not Enough* 165](#_Toc483672903)

[*XSD How To?* 165](#_Toc483672904)

[*A Simple XML Document* 165](#_Toc483672905)

[*A DTD File* 166](#_Toc483672906)

[*An XML Schema* 166](#_Toc483672907)

[*A Reference to a DTD* 166](#_Toc483672908)

[*A Reference to an XML Schema* 167](#_Toc483672909)

[*XSD - The <schema> Element* 167](#_Toc483672910)

[*The <schema> Element* 167](#_Toc483672911)

[*Referencing a Schema in an XML Document* 168](#_Toc483672912)

[*XSD Simple Elements* 169](#_Toc483672913)

[*Defining a Simple Element* 169](#_Toc483672914)

[*Default and Fixed Values for Simple Elements* 170](#_Toc483672915)

[*XSD Attributes* 170](#_Toc483672916)

[*What is an Attribute?* 170](#_Toc483672917)

[*How to Define an Attribute?* 170](#_Toc483672918)

[*Default and Fixed Values for Attributes* 171](#_Toc483672919)

[*Optional and Required Attributes* 171](#_Toc483672920)

[*Restrictions on Content* 171](#_Toc483672921)

[*XSD Restrictions/Facets* 172](#_Toc483672922)

[*Restrictions on Values* 172](#_Toc483672923)

[*Restrictions on a Set of Values* 172](#_Toc483672924)

[*Restrictions on a Series of Values* 173](#_Toc483672925)

[*Other Restrictions on a Series of Values* 174](#_Toc483672926)

[*Restrictions on Whitespace Characters* 175](#_Toc483672927)

[*Restrictions on Length* 176](#_Toc483672928)

[*Restrictions for Datatypes* 176](#_Toc483672929)

[*XSD Complex Elements* 177](#_Toc483672930)

[*What is a Complex Element?* 177](#_Toc483672931)

[*Examples of Complex Elements* 177](#_Toc483672932)

[*How to Define a Complex Element* 178](#_Toc483672933)

[*XSD Empty Elements* 179](#_Toc483672934)

[*Complex Empty Elements* 180](#_Toc483672935)

[*XSD Elements Only* 180](#_Toc483672936)

[*Complex Types Containing Elements Only* 181](#_Toc483672937)

[*XSD Text-Only Elements* 181](#_Toc483672938)

[*Complex Text-Only Elements* 181](#_Toc483672939)

[*XSD Mixed Content* 183](#_Toc483672940)

[*Complex Types with Mixed Content* 183](#_Toc483672941)

[*XSD Indicators* 184](#_Toc483672942)

[*Indicators* 184](#_Toc483672943)

[*Order indicators:* 184](#_Toc483672944)

[*Occurrence indicators:* 184](#_Toc483672945)

[*Group indicators:* 184](#_Toc483672946)

[*Order Indicators* 184](#_Toc483672947)

[*All Indicator* 184](#_Toc483672948)

[*Choice Indicator* 185](#_Toc483672949)

[*Sequence Indicator* 185](#_Toc483672950)

[*Occurrence Indicators* 185](#_Toc483672951)

[*maxOccurs Indicator* 185](#_Toc483672952)

[*minOccurs Indicator* 186](#_Toc483672953)

[*Group Indicators* 187](#_Toc483672954)

[*Element Groups* 187](#_Toc483672955)

[*Attribute Groups* 188](#_Toc483672956)

[*XSD The <any> Element* 189](#_Toc483672957)

[*XSD The <anyAttribute> Element* 190](#_Toc483672958)

[*XSD Element Substitution* 192](#_Toc483672959)

[*Blocking Element Substitution* 192](#_Toc483672960)

[*Using substitutionGroup* 193](#_Toc483672961)

[*What are Global Elements?* 193](#_Toc483672962)

[*An XSD Example* 194](#_Toc483672963)

[*An XML Document* 194](#_Toc483672964)

[*Create an XML Schema* 194](#_Toc483672965)

[*Divide the Schema* 197](#_Toc483672966)

[*Using Named Types* 198](#_Toc483672967)

[*XSD String Data Types* 200](#_Toc483672968)

[*NormalizedString Data Type* 200](#_Toc483672969)

[*Token Data Type* 201](#_Toc483672970)

[*String Data Types* 201](#_Toc483672971)

[*Restrictions on String Data Types* 202](#_Toc483672972)

[*XSD Date and Time Data Types* 202](#_Toc483672973)

[*Date Data Type* 202](#_Toc483672974)

[*Time Zones* 203](#_Toc483672975)

[*Time Data Type* 203](#_Toc483672976)

[*Time Zones* 204](#_Toc483672977)

[*DateTime Data Type* 204](#_Toc483672978)

[*Time Zones* 204](#_Toc483672979)

[*Duration Data Type* 205](#_Toc483672980)

[*Negative Duration* 206](#_Toc483672981)

[*Date and Time Data Types* 206](#_Toc483672982)

[*Restrictions on Date Data Types* 206](#_Toc483672983)

[*XSD Numeric Data Types* 206](#_Toc483672984)

[*Decimal Data Type* 207](#_Toc483672985)

[*Integer Data Type* 207](#_Toc483672986)

[*Numeric Data Types* 208](#_Toc483672987)

[*Restrictions on Numeric Data Types* 208](#_Toc483672988)

[*XSD Miscellaneous Data Types* 209](#_Toc483672989)

[*Boolean Data Type* 209](#_Toc483672990)

[*Binary Data Types* 209](#_Toc483672991)

[*AnyURI Data Type* 209](#_Toc483672992)

[*Miscellaneous Data Types* 209](#_Toc483672993)

[*Restrictions on Miscellaneous Data Types* 210](#_Toc483672994)

[*XML Schema Reference* 210](#_Toc483672995)

[*XSD Elements* 210](#_Toc483672996)

[*XSD Restrictions/Facets for Datatypes* 212](#_Toc483672997)

[XML Web Services 212](#_Toc483672998)

[*XML Web Services* 213](#_Toc483672999)

[*WSDL* 213](#_Toc483673000)

[*SOAP* 213](#_Toc483673001)

[*RDF* 213](#_Toc483673002)

[*RSS* 213](#_Toc483673003)

[*What You Should Already Know* 213](#_Toc483673004)

[*Web Services* 213](#_Toc483673005)

[*Interoperability has Highest Priority* 214](#_Toc483673006)

[*Web Services take Web-applications to the Next Level* 214](#_Toc483673007)

[*Web Services have Two Types of Uses* 214](#_Toc483673008)

[***Reusable application-components.*** 214](#_Toc483673009)

[***Connect existing software.*** 214](#_Toc483673010)

[*Any application can have a Web Service component.* 215](#_Toc483673011)

[*A Web Service Example* 215](#_Toc483673012)

[*Example Explained* 215](#_Toc483673013)

[*Put the Web Service on Your Web Site* 216](#_Toc483673014)

[*How To Do It* 216](#_Toc483673015)

[*XML WSDL* 218](#_Toc483673016)

[*WSDL Documents* 218](#_Toc483673017)

[*WSDL Example* 219](#_Toc483673018)

[*The <portType> Element* 219](#_Toc483673019)

[*WSDL One-Way Operation* 219](#_Toc483673020)

[*WSDL Request-Response Operation* 220](#_Toc483673021)

[*WSDL Binding to SOAP* 221](#_Toc483673022)

[*XML Soap* 222](#_Toc483673023)

[*Why SOAP?* 222](#_Toc483673024)

[*SOAP Building Blocks* 222](#_Toc483673025)

[*Syntax Rules* 222](#_Toc483673026)

[*Skeleton SOAP Message* 223](#_Toc483673027)

[*The SOAP Envelope Element* 223](#_Toc483673028)

[*The xmlns:soap Namespace* 223](#_Toc483673029)

[*The encodingStyle Attribute* 223](#_Toc483673030)

[*A SOAP message has no default encoding.* 224](#_Toc483673031)

[*The SOAP Header Element* 224](#_Toc483673032)

[*The mustUnderstand Attribute* 225](#_Toc483673033)

[*The actor Attribute* 225](#_Toc483673034)

[*The encodingStyle Attribute* 226](#_Toc483673035)

[*The SOAP Body Element* 226](#_Toc483673036)

[*The SOAP Fault Element* 227](#_Toc483673037)

[*SOAP Fault Codes* 227](#_Toc483673038)

[*The HTTP Protocol* 228](#_Toc483673039)

[*SOAP Binding* 228](#_Toc483673040)

[*Content-Type* 229](#_Toc483673041)

[*Content-Length* 229](#_Toc483673042)

[*A SOAP Example* 229](#_Toc483673043)

[*A SOAP request:* 229](#_Toc483673044)

[*The SOAP response:* 230](#_Toc483673045)

[*XML RDF* 230](#_Toc483673046)

[*RDF Document Example* 230](#_Toc483673047)

[*What is RDF?* 230](#_Toc483673048)

[*RDF - Examples of Use* 231](#_Toc483673049)

[*RDF is Designed to be Read by Computers* 231](#_Toc483673050)

[*RDF is Written in XML* 231](#_Toc483673051)

[*RDF and "The Semantic Web"* 231](#_Toc483673052)

[*RDF Resource, Property, and Property Value* 231](#_Toc483673053)

[*RDF Statements* 232](#_Toc483673054)

[*RDF Example* 232](#_Toc483673055)

[*RDF Online Validator* 234](#_Toc483673056)

[*RDF Elements* 234](#_Toc483673057)

[*The <rdf:RDF> Element* 234](#_Toc483673058)

[*The <rdf:Description> Element* 234](#_Toc483673059)

[*Properties as Attributes* 235](#_Toc483673060)

[*Properties as Resources* 235](#_Toc483673061)

[*RDF Containers* 236](#_Toc483673062)

[*The <rdf:Bag> Element* 236](#_Toc483673063)

[*The <rdf:Seq> Element* 236](#_Toc483673064)

[*The <rdf:Alt> Element* 237](#_Toc483673065)

[*RDF Terms* 237](#_Toc483673066)

[*RDF Collections* 238](#_Toc483673067)

[*The rdf:parseType="Collection" Attribute* 238](#_Toc483673068)

[*RDF Schema and Application Classes* 238](#_Toc483673069)

[*RDF Schema (RDFS)* 238](#_Toc483673070)

[*RDFS Example* 239](#_Toc483673071)

[*Example Abbreviated* 239](#_Toc483673072)

[*The Dublin Core* 240](#_Toc483673073)

[*RDF Reference* 241](#_Toc483673074)

[*RDFS / RDF Classes* 241](#_Toc483673075)

[*RDFS / RDF Properties* 242](#_Toc483673076)

[*RDF Attributes* 243](#_Toc483673077)

[*XML RSS* 243](#_Toc483673078)

[*RSS Document Example* 244](#_Toc483673079)

[*What is RSS?* 244](#_Toc483673080)

[*Why use RSS?* 244](#_Toc483673081)

[*Who Should use RSS?* 244](#_Toc483673082)

[*Benefits of RSS* 245](#_Toc483673083)

[***Choose your news*** 245](#_Toc483673084)

[***Remove unwanted information*** 245](#_Toc483673085)

[***Increase your site traffic*** 245](#_Toc483673086)

[*The History of RSS* 245](#_Toc483673087)

[*What RSS Version Should I Use?* 245](#_Toc483673088)

[*Is RSS a Web Standard?* 246](#_Toc483673089)

[*How RSS Works* 246](#_Toc483673090)

[*RSS Example* 246](#_Toc483673091)

[*Comments in RSS* 247](#_Toc483673092)

[*RSS is Written in XML* 247](#_Toc483673093)

[*The RSS <channel> Element* 247](#_Toc483673094)

[*The <category> Element* 248](#_Toc483673095)

[*The <copyright> Element* 248](#_Toc483673096)

[*The <image> Element* 248](#_Toc483673097)

[*The <language> Element* 249](#_Toc483673098)

[*The <item> Element* 249](#_Toc483673099)

[*The <author> Element* 250](#_Toc483673100)

[*The <comments> Element* 250](#_Toc483673101)

[*The <enclosure> Element* 250](#_Toc483673102)

[*Get Your RSS Feed Up On The Web* 250](#_Toc483673103)

[*Can I Manage my RSS Feed Myself?* 251](#_Toc483673104)

[*Automated RSS* 251](#_Toc483673105)

[*RSS Readers* 251](#_Toc483673106)

[*I have an RSS Reader. Now what?* 252](#_Toc483673107)

[*RSS Examples* 252](#_Toc483673108)

[*RSS Reference* 253](#_Toc483673109)

[*RSS <channel> Element* 253](#_Toc483673110)

[*RSS <item> Element* 254](#_Toc483673111)

[*https://www.w3schools.com/xml/default.asp*](https://www.w3schools.com/xml/default.asp)

*XML Tutorial*

*XML stands for eXtensible Markup Language.*

*XML was designed to store and transport data.*

*XML was designed to be both human- and machine-readable.*

*XML Example 1*

*<?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>*

*…*

*Why Study XML?*

*XML plays an important role in many different IT systems.*

*XML is often used for distributing data over the Internet.*

*It is important (for all types of software developers!) to have a good understanding of XML.*

*What You Will Learn*

*This tutorial will give you a solid understanding of XML:*

* *What is XML?*
* *How does XML work?*
* *How can I use XML?*
* *What can I use XML for?*

*Important XML Standards*

*This tutorial will also dig deep into the following important XML standards:*

* [*XML AJAX*](https://www.w3schools.com/xml/ajax_intro.asp)
* [*XML DOM*](https://www.w3schools.com/xml/dom_intro.asp)
* [*XML XPath*](https://www.w3schools.com/xml/xpath_intro.asp)
* [*XML XSLT*](https://www.w3schools.com/xml/xsl_intro.asp)
* [*XML XQuery*](https://www.w3schools.com/xml/xquery_intro.asp)
* [*XML DTD*](https://www.w3schools.com/xml/xml_dtd_intro.asp)
* [*XML Schema*](https://www.w3schools.com/xml/schema_intro.asp)
* [*XML Services*](https://www.w3schools.com/xml/xml_services.asp)

*We recommend reading this tutorial, in the sequence listed in the left menu.*

*Learn by Examples*

*Examples are better than 1000 words. Examples are often easier to understand than text explanations. // важныя аспекты тнав і трэпр //*

*This tutorial supplements all explanations with clarifying "Try it Yourself" examples.*

* [*XML Examples*](https://www.w3schools.com/xml/xml_examples.asp)
* [*AJAX Examples*](https://www.w3schools.com/xml/ajax_examples.asp)
* [*DOM Examples*](https://www.w3schools.com/xml/dom_examples.asp)
* [*XPath Examples*](https://www.w3schools.com/xml/xpath_examples.asp)
* [*XSLT Examples*](https://www.w3schools.com/xml/xsl_examples.asp)

*If you try all the examples, you will learn a lot about XML in a very short time!*

*XML Quiz Test*

[*https://www.w3schools.com/xml/xml\_whatis.asp*](https://www.w3schools.com/xml/xml_whatis.asp)

*Introduction to XML*

*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:*

*…*

*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.*

*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> <data>). 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>*

*…*

*XML Simplifies Things*

* *It simplifies data sharing*
* *It simplifies data transport*
* *It simplifies platform changes*
* *It 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. // праблема абмена дадзенамі між несумясцімымі сістэмамі – вялікая праблема веба //*

*XML stores data in plain text format. This provides a software- and hardware-independent way of storing, transporting, and sharing data. // што значыць шэйрынг дэйта? //*

*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. // не згодны з дадзеным разуменнем чалавека //*

*XML is a W3C Recommendation*

*XML became a W3C Recommendation as early as in February 1998.*

[*https://www.w3schools.com/xml/xml\_usedfor.asp*](https://www.w3schools.com/xml/xml_usedfor.asp)

*How Can XML be Used?*

*XML is used in many aspects of web development.*

*XML is often used to separate data from presentation.*

*XML Separates Data from Presentation*

*XML does not carry any information about how to be displayed. The same XML data can be used in many different presentation scenarios. Because of this, with XML, there is a full separation between data and presentation.*

*XML is Often a Complement to HTML*

*In many HTML applications, XML is used to store or transport data, while HTML is used to format and display the same data.*

*XML Separates Data from HTML*

*When displaying data in HTML, you should not have to edit the HTML file when the data changes. // it is not always so… I think //*

*With XML, the data can be stored in separate XML files. With a few lines of JavaScript code, you can read an XML file and update the data content of any HTML page. // integration of XML, HTML and JavaScript //*

*…*

*You will learn a lot more about using XML and JavaScript in the DOM section of this tutorial.*

*Transaction Data*

*Thousands of XML formats exists, in many different industries, to describe day-to-day data transactions:*

* *Stocks and Shares*
* *Financial transactions*
* *Medical data*
* *Mathematical data*
* *Scientific measurements*
* *News information*
* *Weather services*

*… // тут ідуць два яркіх прыклады захоўвання дадзен //*

[*https://www.w3schools.com/xml/xml\_tree.asp*](https://www.w3schools.com/xml/xml_tree.asp)

*XML Tree*

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

*XML Tree Structure*

**

*…*

*XML Tree Structure*

*XML documents are formed as****element trees****.*

*An XML tree starts at a****root element****and branches from the root to****child elements****.*

*All elements can have sub elements (child elements):*

*<root>  
  <child>  
    <subchild>.....</subchild>  
  </child>  
</root>*

*The terms parent, child, and sibling are used to describe the relationships between elements.*

*Parent have children. Children have parents. Siblings are children on the same level (brothers and sisters). // гэта цудоўная аналогія са структурай сям’і //*

*All elements can have text content (Harry Potter) and attributes (category="cooking").*

*Self-Describing Syntax*

*XML uses a much self-describing syntax.*

*A prolog defines the XML version and the character encoding:*

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

*The next line is the****root element****of the document:*

*<bookstore>*

*The next line starts a <book> element:*

*<book category="cooking">*

*The <book> elements have****4 child elements****: <title>, <author>, <year>, <price>.*

*<title lang="en">Everyday Italian</title>  
<author>Giada De Laurentiis</author>  
<year>2005</year>  
<price>30.00</price>*

*The next line ends the book element:*

*</book>*

*You can assume, from this example, that the XML document contains information about books in a bookstore.*

[*https://www.w3schools.com/xml/xml\_syntax.asp*](https://www.w3schools.com/xml/xml_syntax.asp)

*XML Syntax Rules*

*The syntax rules of XML are very simple and logical. The rules are easy to learn, and easy to use.*

*XML Documents Must Have a Root Element*

*XML documents must contain one****root****element that is the****parent****of all other elements:*

*<root>  
  <child>  
    <subchild>.....</subchild>  
  </child>  
</root>*

*The XML Prolog*

*This line is called the XML****prolog****:*

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

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

*XML documents can contain international characters, like Norwegian øæå or French êèé. To avoid errors, you should specify the encoding used, or save your XML files as UTF-8.*

*UTF-8 is the default character encoding for XML documents. // а як пры захаванні вызначаць кадзіроўку? //*

*Character encoding can be studied in our*[*Character Set Tutorial*](https://www.w3schools.com/charsets/default.asp)*. // !!!!!!!!!! //*

*UTF-8 is also the default encoding for HTML5, CSS, JavaScript, PHP, and SQL.*

*All XML Elements Must Have a Closing Tag*

*In HTML, some elements might work well, even with a missing closing tag:*

*<p>This is a paragraph.  
<br>*

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

*<p>This is a paragraph.</p>  
<br />*

*The XML prolog does not have a closing tag.  
This is not an error. The prolog is not a part of the XML document.*

*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 incorrect</message>  
<message>This is correct</message>*

*"Opening and closing tags" are often referred to as "Start and end tags". Use whatever you prefer. It is exactly the same thing.*

*XML Elements Must be Properly Nested*

*In HTML, you might see improperly nested elements:*

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

*In XML, all elements****must****be properly nested within each other:*

*<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.*

*XML Attribute Values Must be Quoted*

*XML elements can have attributes in name/value pairs just like in HTML.*

*In XML, the attribute values must always be quoted.*

*INCORRECT:*

*<note date=12/11/2007>  
  <to>Tove</to>  
  <from>Jani</from>  
</note>*

*CORRECT:*

*<note date="12/11/2007">  
  <to>Tove</to>  
  <from>Jani</from>  
</note>*

*The error in the first document is that the date attribute in the note element is not quoted.*

*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* |

*Only < and & are strictly illegal in XML, but it is a good habit to replace > with &gt; as well.*

*Comments in XML*

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

*<!-- This is a comment -->*

*Two dashes in the middle of a comment are not allowed.*

*Not allowed:*

*<!-- This is a -- comment -->*

*Strange, but allowed:*

*<!-- 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): // а ў спецыфікацыі, здаецца, узгадвалася, што гэта не так… //*

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

*XML Stores New Line as LF*

*Windows applications store a new line as: carriage return and line feed (CR+LF). Unix and Mac OSX uses LF. Old Mac systems uses CR. XML stores a new line as LF.*

*Well Formed XML*

*XML documents that conform to the syntax rules above are said to be "Well Formed" XML documents. // concept definition //*

[*https://www.w3schools.com/xml/xml\_elements.asp*](https://www.w3schools.com/xml/xml_elements.asp)

*XML Elements*

*An XML document contains XML Elements.*

*What is an XML Element?*

*An XML element is everything from (including) the element's start tag to (including) the element's end tag.*

*<price>29.99</price>*

*An element can contain:*

* *text*
* *attributes*
* *other elements*
* *or a mix of the above*

*<bookstore>  
  <book category="children">  
    <title>Harry Potter</title>  
    <author>J K. Rowling</author>  
    <year>2005</year>  
    <price>29.99</price>  
  </book>  
  <book category="web">  
    <title>Learning XML</title>  
    <author>Erik T. Ray</author>  
    <year>2003</year>  
    <price>39.95</price>  
  </book>  
</bookstore>*

*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").*

*Empty XML Elements*

*An element with no content is said to be empty.*

*In XML, you can indicate an empty element like this:*

*<element></element>*

*You can also use a so called self-closing tag:*

*<element />*

*The two forms produce identical results in XML software (Readers, Parsers, Browsers).*

*Empty elements can have attributes.*

*XML Naming Rules*

*XML elements must follow these naming rules:*

* *Element names are case-sensitive*
* *Element names must start with a letter or underscore*
* *Element names cannot start with the letters xml (or XML, or Xml, etc)*
* *Element names can contain letters, digits, hyphens, underscores, and periods*
* *Element names cannot contain spaces*

*Any name can be used, no words are reserved (except xml).*

*Best Naming Practices*

*Create descriptive names, like this: <person>, <firstname>, <lastname>.*

*Create short and simple names, like this: <book\_title> not like this: <the\_title\_of\_the\_book>.*

*Avoid "-". If you name something "first-name", some software may think you want to subtract "name" from "first".*

*Avoid ".". If you name something "first.name", some software may think that "name" is a property of the object "first".*

*Avoid ":". Colons are reserved for namespaces (more later).*

*Non-English letters like éòá are perfectly legal in XML, but watch out for problems if your software doesn't support them.*

*Naming Styles*

*There are no naming styles defined for XML elements. But here are some commonly used:*

|  |  |  |
| --- | --- | --- |
| ***Style*** | ***Example*** | ***Description*** |
| *Lower case* | *<firstname>* | *All letters lower case* |
| *Upper case* | *<FIRSTNAME>* | *All letters upper case* |
| *Underscore* | *<first\_name>* | *Underscore separates words* |
| *Pascal case* | *<FirstName>* | *Uppercase first letter in each word* |
| *Camel case* | *<firstName>* | *Uppercase first letter in each word except the first* |

*If you choose a naming style, it is good to be consistent!*

*XML documents often have a corresponding database. A common practice is to use the naming rules of the database for the XML elements. // тыповая арганізацыя інфармацыйных сістэм. Відаць, xml выкарыстоўваюць для перадачы дадзен, а базы – для захавання. Паразважаць. //*

*Camel case is a common naming rule in JavaScripts.*

*XML Elements are Extensible*

*XML elements can be extended to carry more information.*

*Look at the following XML example:*

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

*Let's imagine that we created an application that extracted the <to>, <from>, and <body> elements from the XML document to produce this output:*

|  |
| --- |
| ***MESSAGE***  ***To:****Tove* ***From:****Jani*  *Don't forget me this weekend!* |

*Imagine that the author of the XML document added some extra information to it:*

*<note>  
  <date>2008-01-10</date>  
  <to>Tove</to>  
  <from>Jani</from>  
  <heading>Reminder</heading>  
  <body>Don't forget me this weekend!</body>  
</note>*

*Should the application break or crash?*

*No. The application should still be able to find the <to>, <from>, and <body> elements in the XML document and produce the same output.*

*This is one of the beauties of XML. It can be extended without breaking applications.*

[*https://www.w3schools.com/xml/xml\_attributes.asp*](https://www.w3schools.com/xml/xml_attributes.asp)

*XML Attributes*

*XML elements can have attributes, just like HTML.*

*Attributes are designed to contain data related to a specific element.*

*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'>*

*If the attribute value itself contains double quotes you can use single quotes, like in this example:*

*<gangster name='George "Shotgun" Ziegler'>*

*or you can use character entities:*

*<gangster name="George &quot;Shotgun&quot; Ziegler">*

*XML Elements vs. Attributes*

*Take a look at these examples:*

*<person gender="female">  
  <firstname>Anna</firstname>  
  <lastname>Smith</lastname>  
</person>*

*<person>  
  <gender>female</gender>  
  <firstname>Anna</firstname>  
  <lastname>Smith</lastname>  
</person>*

*In the first example gender is an attribute. In the last, gender is an element. Both examples provide the same information. There are no rules about when to use attributes or when to use elements in XML.*

*My Favorite Way*

*The following three XML documents contain exactly the same information:*

*A date attribute is used in the first example:*

*<note date="2008-01-10">  
  <to>Tove</to>  
  <from>Jani</from>  
</note>*

*A <date> element is used in the second example:*

*<note>  
  <date>2008-01-10</date>  
  <to>Tove</to>  
  <from>Jani</from>  
</note>*

*An expanded <date> element is used in the third example: (THIS IS MY FAVORITE):*

*<note>  
  <date>  
    <year>2008</year>  
    <month>01</month>  
    <day>10</day>  
  </date>  
  <to>Tove</to>  
  <from>Jani</from>  
</note>*

*Avoid XML Attributes?*

*Some things to consider when using attributes are:*

* *attributes cannot contain multiple values (elements can)*
* *attributes cannot contain tree structures (elements can)*
* *attributes are not easily expandable (for future changes) // intbu examples //*

*Don't end up like this:*

*<note day="10" month="01" year="2008"  
to="Tove" from="Jani" heading="Reminder"  
body="Don't forget me this weekend!">  
</note>*

*XML Attributes for Metadata*

*Sometimes ID references are assigned to elements. These IDs can be used to identify XML elements in much the same way as the id attribute in HTML. This example demonstrates this:*

*…*

*The id attributes above are for identifying the different notes. It is not a part of the note itself.*

*What I'm trying to say here is that metadata (data about data) should be stored as attributes, and the data itself should be stored as elements.*

[*https://www.w3schools.com/xml/xml\_namespaces.asp*](https://www.w3schools.com/xml/xml_namespaces.asp)

*XML Namespaces*

*XML Namespaces provide a method to avoid element name conflicts.*

*Name Conflicts*

*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.*

*This XML carries HTML table information:*

*<table>  
  <tr>  
    <td>Apples</td>  
    <td>Bananas</td>  
  </tr>  
</table>*

*This XML carries information about a table (a piece of furniture):*

*<table>  
  <name>African Coffee Table</name>  
  <width>80</width>  
  <length>120</length>  
</table>*

*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.*

*A user or an XML application will not know how to handle these differences.*

*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:*

*<h:table>  
  <h:tr>  
    <h:td>Apples</h:td>  
    <h:td>Bananas</h:td>  
  </h:tr>  
</h:table>  
  
<f:table>  
  <f:name>African Coffee Table</f:name>  
  <f:width>80</f:width>  
  <f:length>120</f:length>  
</f:table>*

*In the example above, there will be no conflict because the two <table> elements have different names.*

*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".*

*<root>  
<h:table xmlns:h="http://www.w3.org/TR/html4/">  
  <h:tr>  
    <h:td>Apples</h:td>  
    <h:td>Bananas</h:td>  
  </h:tr>  
</h:table>  
  
<f:table xmlns:f="https://www.w3schools.com/furniture">  
  <f:name>African Coffee Table</f:name>  
  <f:width>80</f:width>  
  <f:length>120</f:length>  
</f:table>  
  
</root>*

*In the example above:*

*The xmlns attribute in the first <table> element gives the h: prefix a qualified namespace.*

*The xmlns attribute in the second <table> element gives the f: prefix a qualified namespace.*

*When a namespace is defined for an element, all child elements with the same prefix are associated with the same namespace.*

*Namespaces can also be declared in the XML root element:*

*<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:td>Bananas</h:td>  
  </h:tr>  
</h:table>  
  
<f:table>  
  <f:name>African Coffee Table</f:name>  
  <f:width>80</f:width>  
  <f:length>120</f:length>  
</f:table>  
  
</root>*

***Note:****The namespace URI is not used by the parser to look up information.*

*The purpose of using an URI is to give the namespace a unique name. However, companies often use the namespace as a pointer to a web page containing namespace information.*

*Uniform Resource Identifier (URI)*

*A****Uniform Resource Identifier****(URI) is a string of characters which identifies an Internet Resource.*

*The most common URI is the****Uniform Resource Locator****(URL) which identifies an Internet domain address. Another, not so common type of URI is the****Universal Resource Name****(URN).*

*Default Namespaces*

*Defining a default namespace for an element saves us from using prefixes in all the child elements. It has the following syntax: xmlns="namespaceURI"*

*This XML carries HTML table information:*

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

*This XML carries information about a piece of furniture:*

*<table xmlns="https://www.w3schools.com/furniture">  
  <name>African Coffee Table</name>  
  <width>80</width>  
  <length>120</length>  
</table>*

*Namespaces in Real Use*

*XSLT is a language that can be used to transform XML documents into other formats.*

*The XML document below, is a document used to transform XML into HTML.*

*The namespace "http://www.w3.org/1999/XSL/Transform" identifies XSLT elements inside an HTML document:*

*… // я не вельмі гэты прыклад зразумеў… //*

*If you want to learn more about XSLT, please read our*[*XSLT Tutorial*](https://www.w3schools.com/xml/xsl_intro.asp)*.*

[*https://www.w3schools.com/xml/xml\_display.asp*](https://www.w3schools.com/xml/xml_display.asp)

*Displaying XML*

*Raw XML files can be viewed in all major browsers.*

*Don't expect XML files to be displayed as HTML pages.*

*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.*

***Note:****In Safari 5 (and earlier), only the element text will be displayed. To view the raw XML, you must right click the page and select "View Source".*

*Viewing an Invalid XML File*

*If an erroneous XML file is opened, some browsers will report the error, and some will display it, or display it incorrectly.*

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

*Try to open the following XML file:*[*note\_error.xml*](javascript:void(0))

*Other XML Examples*

*Viewing some XML documents will help you get the XML feeling:*

*…*

*Why Does XML Display Like This?*

*XML documents do not carry information about how to display the data.*

*Since XML tags are "invented" by the author of the XML document, browsers do not know if a tag like <table> describes an HTML table or a dining table.*

*Without any information about how to display the data, the browsers can just display the XML document as it is.*

*Displaying XML Files with CSS?*

*Below is an example of how to use CSS to format an XML document.*

*We can use an XML file like*[*cd\_catalog.xml*](javascript:void(0))*and a style sheet like*[*cd\_catalog.css*](https://www.w3schools.com/xml/cd_catalog.css)

*…*

*Formatting XML with CSS is not recommended. Use JavaScript or XSLT instead.*

[*https://www.w3schools.com/xml/xml\_http.asp*](https://www.w3schools.com/xml/xml_http.asp)

*XML HttpRequest*

*All modern browsers have a built-in XMLHttpRequest object to request data from a server. // intbu programmly, physically. Logical meaning I understood. //*

*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*

*XMLHttpRequest Example*

*When you type a character in the input field below, an XMLHttpRequest is sent to the server, and some name suggestions are returned (from the server):*

*…*

*The example above is explained in the AJAX chapters of this tutorial.*

*Sending an XMLHttpRequest*

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

*Example*

*var xhttp = new XMLHttpRequest();  
xhttp.onreadystatechange = function() {  
    if (this.readyState == 4 && this.status == 200) {  
       // Typical action to be performed when the document is ready:  
       document.getElementById("demo").innerHTML = xhttp.responseText;  
    }  
};  
xhttp.open("GET", "filename", true);  
xhttp.send();*

*Example Explained*

*The first line in the example above creates an****XMLHttpRequest****object:*

*var xhttp = new XMLHttpRequest();*

*The****onreadystatechange****property specifies a function to be executed every time the status of the XMLHttpRequest object changes:*

*xhttp.onreadystatechange = function()*

*When****readyState****property is 4 and the****status****property is 200, the response is ready:*

*if (this.readyState == 4 && this.status == 200)*

*The****responseText****property returns the server response as a text string.*

*The text string can be used to update a web page:*

*document.getElementById("demo").innerHTML = xhttp.responseText;*

*You will learn a lot more about the XMLHttpRequest object in the AJAX chapters of this tutorial.*

*Old Versions of Internet Explorer (IE5 and IE6)*

*…*

[*https://www.w3schools.com/xml/xml\_parser.asp*](https://www.w3schools.com/xml/xml_parser.asp)

*XML Parser*

*All major browsers have a built-in XML parser to access and manipulate XML.*

*XML Parser*

*The*[*XML DOM (Document Object Model)*](https://www.w3schools.com/xml/dom_intro.asp)*defines the properties and methods for accessing and editing XML. However, before an XML document can be accessed, it must be loaded into an XML DOM object.*

*All modern browsers have a built-in XML parser that can convert text into an XML DOM object.*

*Parsing a Text String*

*This example parses a text string into an XML DOM object, and extracts the info from it with JavaScript:*

*Example*

*<html>  
<body>  
  
<p id="demo"></p>  
  
<script>  
var text, parser, xmlDoc;  
  
text = "<bookstore><book>" +  
"<title>Everyday Italian</title>" +  
"<author>Giada De Laurentiis</author>" +  
"<year>2005</year>" +  
"</book></bookstore>";  
  
parser = new DOMParser();  
xmlDoc = parser.parseFromString(text,"text/xml");  
  
document.getElementById("demo").innerHTML =  
xmlDoc.getElementsByTagName("title")[0].childNodes[0].nodeValue;  
</script>  
  
</body>  
</html>*

*// прыклад паспрабаваў. Атрымалася трошку памацаць метады JavaScript //*

*Example Explained*

*A text string is defined:*

*text = "<bookstore><book>" +  
"<title>Everyday Italian</title>" +  
"<author>Giada De Laurentiis</author>" +  
"<year>2005</year>" +  
"</book></bookstore>";*

*An XML DOM parser is created:*

*parser = new DOMParser();*

*The parser creates a new XML DOM object using the text string:*

*xmlDoc = parser.parseFromString(text,"text/xml");*

*Old Versions of Internet Explorer*

*…*

*The XMLHttpRequest Object*

*The*[*XMLHttpRequest Object*](https://www.w3schools.com/xml/xml_http.asp)*has a built in XML Parser. The****responseText****property returns the response as a string. The****responseXML****property returns the response as an XML DOM object. If you want to use the response as an XML DOM object, you can use the responseXML property.*

*Example*

*Request the file*[*cd\_catalog.xml*](https://www.w3schools.com/xml/cd_catalog.xml)*and use the response as an XML DOM object:*

*xmlDoc = xmlhttp.responseXML;  
txt = "";  
x = xmlDoc.getElementsByTagName("ARTIST");  
for (i = 0; i < x.length; i++) {  
    txt += x[i].childNodes[0].nodeValue + "<br>";  
}  
document.getElementById("demo").innerHTML = txt;*

*// у гэтым прыкладзе добра бачна, як можна выкарыстоўваць разам xml, JavaScript і html //*

[*https://www.w3schools.com/xml/xml\_dom.asp*](https://www.w3schools.com/xml/xml_dom.asp)

*XML DOM*

**

*What is the DOM?*

*The DOM defines a standard for accessing and manipulating documents:*

*"The W3C Document Object Model (DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document."*

*The HTML DOM defines a standard way for accessing and manipulating HTML documents. It presents an HTML document as a tree-structure.*

*The XML DOM defines a standard way for accessing and manipulating XML documents. It presents an XML document as a tree-structure.*

*Understanding the DOM is a must for anyone working with HTML or XML.*

*The HTML DOM*

*All HTML elements can be accessed through the HTML DOM. This example changes the value of an HTML element with id="demo":*

*Example*

*<h1 id="demo">This is a Heading</h1>  
  
<button type="button"  
onclick="document.getElementById('demo').innerHTML = 'Hello World!'">Click Me!  
</button>*

[*Try it Yourself »*](https://www.w3schools.com/xml/tryit.asp?filename=tryxml_change_html) *// прыклад цудоўна паспрабаваў. Усё працуе //*

*You can learn a lot more about the HTML DOM in our*[*JavaScript tutorial*](https://www.w3schools.com/js/js_htmldom.asp)*.*

*The XML DOM*

*All XML elements can be accessed through the XML DOM.*

*Books.xml*

*<?xml version="1.0" encoding="UTF-8"?>  
<bookstore>  
  
  <book category="cooking">  
    <title lang="en">Everyday Italian</title>  
    <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 code retrieves the text value of the first <title> element in an XML document:*

*Example*

*txt = xmlDoc.getElementsByTagName("title")[0].childNodes[0].nodeValue;*

*The XML DOM is a standard for how to get, change, add, and delete XML elements.*

*This example loads a text string into an XML DOM object, and extracts the info from it with JavaScript:*

*Example*

*<html>  
<body>  
  
<p id="demo"></p>  
  
<script>  
var text, parser, xmlDoc;  
  
text = "<bookstore><book>" +  
"<title>Everyday Italian</title>" +  
"<author>Giada De Laurentiis</author>" +  
"<year>2005</year>" +  
"</book></bookstore>";  
  
parser = new DOMParser();  
xmlDoc = parser.parseFromString(text,"text/xml");  
  
document.getElementById("demo").innerHTML =****xmlDoc.getElementsByTagName("title")[0].childNodes[0].nodeValue****;  
</script>  
  
</body>  
</html>*

[*Try it Yourself »*](https://www.w3schools.com/xml/tryit.asp?filename=try_dom_loadxmltext) *// паспрабаваў прыклад. Вельмі наглядны і ясны //*

*You will learn a lot more about the XML DOM in our*[*XML DOM Tutorial*](https://www.w3schools.com/xml/dom_intro.asp)*.*

[*https://www.w3schools.com/xml/xml\_xpath.asp*](https://www.w3schools.com/xml/xml_xpath.asp)

*XML and XPath*

*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* | * *XPath is a syntax for defining parts of an XML document* * *XPath uses path expressions to navigate in XML documents* * *XPath contains a library of standard functions* * *XPath is a major element in XSLT and in XQuery* * *XPath is a W3C recommendation* |

*XPath Path Expressions*

*XPath uses path expressions to select nodes or node-sets in an XML document. These path expressions look very much like the expressions you see when you work with a traditional computer file system.*

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

*XPath is Used in XSLT*

*XPath is a major element in the XSLT standard. With XPath knowledge you will be able to take great advantage of XSL.*

*XPath Example*

*We will use the following XML document:*

*<?xml version="1.0" encoding="UTF-8"?>  
  
<bookstore>  
  
<book category="cooking">  
  <title lang="en">Everyday Italian</title>  
  <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>  
  
<book category="web">  
  <title lang="en">XQuery Kick Start</title>  
  <author>James McGovern</author>  
  <author>Per Bothner</author>  
  <author>Kurt Cagle</author>  
  <author>James Linn</author>  
  <author>Vaidyanathan Nagarajan</author>  
  <year>2003</year>  
  <price>49.99</price>  
</book>  
  
<book category="web">  
  <title lang="en">Learning XML</title>  
  <author>Erik T. Ray</author>  
  <year>2003</year>  
  <price>39.95</price>  
</book>  
  
</bookstore>*

*In the table below we have listed some XPath expressions and the result of the expressions:*

|  |  |
| --- | --- |
| ***XPath Expression*** | ***Result*** |
| */bookstore/book[1]* | *Selects the first book element that is the child of the bookstore element* |
| */bookstore/book[last()]* | *Selects the last book element that is the child of the bookstore element* |
| */bookstore/book[last()-1]* | *Selects the last but one book element that is the child of the bookstore element* |
| */bookstore/book[position()<3]* | *Selects the first two book elements that are children of the bookstore element* |
| *//title[@lang]* | *Selects all the title elements that have an attribute named lang* |
| *//title[@lang='en']* | *Selects all the title elements that have a "lang" attribute with a value of "en"* |
| */bookstore/book[price>35.00]* | *Selects all the book elements of the bookstore element that have a price element with a value greater than 35.00* |
| */bookstore/book[price>35.00]/title* | *Selects all the title elements of the book elements of the bookstore element that have a price element with a value greater than 35.00* |

*// xpath можна выявіць як набор селектараў. Паразважаць. //*

*You will learn a lot more about XPath in our*[*XPath Tutorial*](https://www.w3schools.com/xml/xpath_intro.asp)*.*

[*https://www.w3schools.com/xml/xml\_xslt.asp*](https://www.w3schools.com/xml/xml_xslt.asp)

*XML and XSLT*

*With XSLT you can transform an XML document into HTML.*

*Displaying XML with XSLT*

*XSLT (eXtensible Stylesheet Language Transformations) is the recommended style sheet language for XML.*

*XSLT is far more sophisticated than CSS. With XSLT you can add/remove elements and attributes to or from the output file. You can also rearrange and sort elements, perform tests and make decisions about which elements to hide and display, and a lot more.*

*XSLT uses XPath to find information in an XML document.*

*XSLT Example*

*…*

*Use XSLT to transform XML into HTML, before it is displayed in a browser:*

*Example XSLT Stylesheet:*

*…*

*If you want to learn more about XSLT, go to our*[*XSLT Tutorial*](https://www.w3schools.com/xml/xsl_intro.asp)*.*

[*https://www.w3schools.com/xml/xml\_xquery.asp*](https://www.w3schools.com/xml/xml_xquery.asp)

*XML and XQuery*

*What is XQuery?*

*XQuery is to XML what SQL is to databases.*

*XQuery was designed to query XML data.*

*XQuery Example*

*for $x in doc("books.xml")/bookstore/book  
where $x/price>30  
order by $x/title  
return $x/title*

*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 is About Querying XML*

*XQuery is a language for finding and extracting elements and attributes from XML documents.*

*Here is an example of what XQuery could solve:*

*"Select all CD records with a price less than $10 from the CD collection stored in cd\_catalog.xml"*

*XQuery and XPath*

*XQuery 1.0 and XPath 2.0 share the same data model and support the same functions and operators. If you have already studied XPath you will have no problems with understanding XQuery.*

*XQuery - Examples of Use*

*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*

*// intbu first three purposes //*

*XQuery is a W3C Recommendation*

*XQuery is compatible with several W3C standards, such as XML, Namespaces, XSLT, XPath, and XML Schema. XQuery 1.0 became a W3C Recommendation in 2007.*

*You will learn a lot more about XQuery in our*[*XQuery Tutorial*](https://www.w3schools.com/xml/xquery_intro.asp)*.*

[*https://www.w3schools.com/xml/xml\_xlink.asp*](https://www.w3schools.com/xml/xml_xlink.asp)

*XML, XLink and XPointer*

*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* |

*XLink Browser Support*

*There is no browser support for XLink in XML documents. However, all major browsers support* [*XLinks in SVG*](https://www.w3schools.com/graphics/svg_text.asp)*. // and what is necessary to do? //*

*XLink Syntax*

*In HTML, the <a> element defines a hyperlink. However, this is not how it works in XML. In XML documents, you can use whatever element names you want - therefore it is impossible for browsers to predict what link elements will be called in XML documents. // difficulty to define hyperlink in xml //*

*Below is a simple example of how to use XLink to create links in an XML document:*

*<?xml version="1.0" encoding="UTF-8"?>  
  
<homepages xmlns:xlink="http://www.w3.org/1999/xlink">  
  <homepage xlink:type="simple" xlink:href="https://www.w3schools.com">Visit W3Schools</homepage>  
  <homepage xlink:type="simple" xlink:href="http://www.w3.org">Visit W3C</homepage>  
</homepages>*

*To get access to the XLink features we must declare the XLink namespace. The XLink namespace is: "http://www.w3.org/1999/xlink".*

*The xlink:type and the xlink:href attributes in the <homepage> elements come from the XLink namespace.*

*The xlink:type="simple" creates a simple "HTML-like" link (means "click here to go there").*

*The xlink:href attribute specifies the URL to link to.*

*XLink Example*

*The following XML document contains XLink features:*

*<?xml version="1.0" encoding="UTF-8"?>  
  
<bookstore xmlns:xlink="http://www.w3.org/1999/xlink">  
  
<book title="Harry Potter">  
  <description  
  xlink:type="simple"  
  xlink:href="/images/HPotter.gif"  
  xlink:show="new">  
  As his fifth year at Hogwarts School of Witchcraft and  
  Wizardry approaches, 15-year-old Harry Potter is.......  
  </description>  
</book>  
  
<book title="XQuery Kick Start">  
  <description  
  xlink:type="simple"  
  xlink:href="/images/XQuery.gif"  
  xlink:show="new">  
  XQuery Kick Start delivers a concise introduction  
  to the XQuery standard.......  
  </description>  
</book>  
  
</bookstore>*

***Example explained:***

* *The XLink namespace is declared at the top of the document (xmlns:xlink="http://www.w3.org/1999/xlink")*
* *The xlink:type="simple" creates a simple "HTML-like" link*
* *The xlink:href attribute specifies the URL to link to (in this case - an image)*
* *The xlink:show="new" specifies that the link should open in a new window*

*// XLind examplanation in example //*

*XLink - Going Further*

*In the example above we have demonstrated simple XLinks. XLink is getting more interesting when accessing remote locations as resources, instead of standalone pages.*

*If we set the value of the xlink:show attribute to "embed", the linked resource should be processed inline within the page. When you consider that this could be another XML document you could, for example, build a hierarchy of XML documents.*

*You can also specify WHEN the resource should appear, with the xlink:actuate attribute.*

*XLink Attribute Reference*

|  |  |  |
| --- | --- | --- |
| ***Attribute*** | ***Value*** | ***Description*** |
| *xlink:actuate* | *onLoad onRequest other none* | *Defines when the linked resource is read and shown:*   * *onLoad - the resource should be loaded and shown when the document loads* * *onRequest - the resource is not read or shown before the link is clicked* |
| *xlink:href* | *URL* | *Specifies the URL to link to* |
| *xlink:show* | *embed new replace other none* | *Specifies where to open the link. Default is "replace"* |
| *xlink:type* | *simple extended locator arc resource title none* | *Specifies the type of link* |

*// intbu this language //*

*XPointer*

|  |  |
| --- | --- |
| *XPath* | * *XPointer allows links to point to specific parts of an XML document* * *XPointer uses XPath expressions to navigate in the XML document* * *XPointer is a W3C Recommendation* |

*XPointer Browser Support*

*There is no browser support for XPointer. But XPointer is used in other XML languages. // what are xml-languages? //*

*XPointer Example*

*In this example, we will use XPointer in conjunction with XLink to point to a specific part of another document. We will start by looking at the target XML document (the document we are linking to):*

*<?xml version="1.0" encoding="UTF-8"?>  
  
<dogbreeds>  
  
<dog breed="Rottweiler" id="Rottweiler">  
  <picture url="https://dog.com/rottweiler.gif" />  
  <history>The Rottweiler's ancestors were probably Roman  
  drover dogs.....</history>  
  <temperament>Confident, bold, alert and imposing, the Rottweiler  
  is a popular choice for its ability to protect....</temperament>  
</dog>  
  
<dog breed="FCRetriever" id="FCRetriever">  
  <picture url="https://dog.com/fcretriever.gif" />  
  <history>One of the earliest uses of retrieving dogs was to  
  help fishermen retrieve fish from the water....</history>  
  <temperament>The flat-coated retriever is a sweet, exuberant,  
  lively dog that loves to play and retrieve....</temperament>  
</dog>  
  
</dogbreeds>*

*Note that the XML document above uses id attributes on each element!*

*So, instead of linking to the entire document (as with XLink), XPointer allows you to link to specific parts of the document. To link to a specific part of a page, add a number sign (#) and an XPointer expression after the URL in the xlink:href attribute, like this: xlink:href="https://dog.com/dogbreeds.xml#xpointer(id('Rottweiler'))". The expression refers to the element in the target document, with the id value of "Rottweiler".*

*XPointer also allows a shorthand method for linking to an element with an id. You can use the value of the id directly, like this: xlink:href="https://dog.com/dogbreeds.xml#Rottweiler".*

*The following XML document contains links to more information of the dog breed for each of my dogs:*

*<?xml version="1.0" encoding="UTF-8"?>  
  
<mydogs xmlns:xlink="http://www.w3.org/1999/xlink">  
  
<mydog>  
  <description>  
  Anton is my favorite dog. He has won a lot of.....  
  </description>  
  <fact xlink:type="simple" xlink:href="https://dog.com/dogbreeds.xml#Rottweiler">  
  Fact about Rottweiler   
  </fact>  
</mydog>  
  
<mydog>  
  <description>  
  Pluto is the sweetest dog on earth......  
  </description>  
  <fact xlink:type="simple" xlink:href="https://dog.com/dogbreeds.xml#FCRetriever">  
  Fact about flat-coated Retriever   
  </fact>  
</mydog>  
  
</mydogs>*

*// тут дададзены спасылкі на інфармацыю аб пародах сабак //*

[*https://www.w3schools.com/xml/xml\_validator.asp*](https://www.w3schools.com/xml/xml_validator.asp)

*XML Validator*

*Use our XML validator to syntax-check your XML.*

*Well Formed XML Documents*

*An XML document with correct syntax is called "Well Formed".*

*The syntax rules were described in the previous chapters:*

* *XML documents must have a root element*
* *XML elements must have a closing tag*
* *XML tags are case sensitive*
* *XML elements must be properly nested*
* *XML attribute values must be quoted*

*<?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>*

*XML Errors Will Stop You*

*Errors in XML documents will stop your XML applications. The W3C XML specification states that a program should stop processing an XML document if it finds an error. // дзе гэта было такое сказана? // The reason is that XML software should be small, fast, and compatible. // intbu the reason //*

*HTML browsers are allowed to display HTML documents with errors (like missing end tags).*

***With XML, errors are not allowed.***

*Syntax-Check Your XML*

*To help you syntax-check your XML, we have created an XML validator.*

*Try to syntax-check correct XML : … // validator is on this page //*

*Valid XML Documents*

*A "well formed" XML document is not the same as a "valid" XML document.*

*A "valid" XML document must be well formed. In addition, it must conform to a document type definition.*

*There are two different document type definitions that can be used with XML:*

* *DTD - The original Document Type Definition*
* *XML Schema - An XML-based alternative to DTD*

*A document type definition defines the rules and the legal elements and attributes for an XML document.*

[*https://www.w3schools.com/xml/xml\_dtd.asp*](https://www.w3schools.com/xml/xml_dtd.asp)

*XML DTD*

*An XML document with correct syntax is called "Well Formed".*

*An XML document validated against a DTD is both "Well Formed" and "Valid".*

*Valid XML Documents*

*A "Valid" XML document is a "Well Formed" XML document, which also conforms to the rules of a DTD:*

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

*The DOCTYPE declaration, in the example above, is a reference to an external DTD file. The content of the file is shown in the paragraph below.*

*XML DTD*

*The purpose of a DTD is to define the structure of an XML document. It defines the structure with a list of legal elements:*

*<!DOCTYPE note  
[  
<!ELEMENT note (to,from,heading,body)>  
<!ELEMENT to (#PCDATA)>  
<!ELEMENT from (#PCDATA)>  
<!ELEMENT heading (#PCDATA)>  
<!ELEMENT body (#PCDATA)>  
]>*

*The DTD above is interpreted like this:*

* *!DOCTYPE note defines that the root element of the document is note*
* *!ELEMENT note defines that the note element must contain the elements: "to, from, heading, body"*
* *!ELEMENT to defines the to element to be of type "#PCDATA"*
* *!ELEMENT from defines the from element to be of type "#PCDATA"*
* *!ELEMENT heading defines the heading element to be of type "#PCDATA"*
* *!ELEMENT body defines the body element to be of type "#PCDATA"*

*#PCDATA means parse-able text data.*

*Using DTD for Entity Declaration*

*A doctype declaration can also be used to define special characters and character strings, used in the document:*

*Example*

*<?xml version="1.0" encoding="UTF-8"?>  
  
<!DOCTYPE note [  
<!ENTITY nbsp "&#xA0;">   
<!ENTITY writer "Writer: Donald Duck.">  
<!ENTITY copyright "Copyright: W3Schools.">  
]>  
  
<note>  
<to>Tove</to>  
<from>Jani</from>  
<heading>Reminder</heading>  
<body>Don't forget me this weekend!</body>  
<footer>&writer;&nbsp;&copyright;</footer>  
</note>*

*// нарэшце я зразумеў, што такое энціці //*

*An entity has three parts: an ampersand (&), an entity name, and a semicolon (;).*

*When to Use a DTD/Schema?*

*With a DTD, independent groups of people can agree to use a standard DTD for interchanging data.*

*With a DTD, you can verify that the data you receive from the outside world is valid.*

*You can also use a DTD to verify your own data.*

*If you want to study DTD, please read our*[*DTD Tutorial*](https://www.w3schools.com/xml/xml_dtd_intro.asp)*.*

*When to NOT to Use a DTD/Schema?*

*XML does not require a DTD/Schema.*

*When you are experimenting with XML, or when you are working with small XML files, creating DTDs may be a waste of time.*

*If you develop applications, wait until the specification is stable before you add a document definition. Otherwise, your software might stop working because of validation errors.*

*// вельмі-вельмі каштоўная парада //*

[*https://www.w3schools.com/xml/xml\_schema.asp*](https://www.w3schools.com/xml/xml_schema.asp)

*XML Schema*

*An XML Schema describes the structure of an XML document, just like a DTD.*

*An XML document with correct syntax is called "Well Formed".*

*An XML document validated against an XML Schema is both "Well Formed" and "Valid".*

*XML Schema*

*XML Schema is an XML-based alternative to DTD:*

*<xs:element name="note">  
  
<xs:complexType>  
  <xs:sequence>  
    <xs:element name="to" type="xs:string"/>  
    <xs:element name="from" type="xs:string"/>  
    <xs:element name="heading" type="xs:string"/>  
    <xs:element name="body" type="xs:string"/>  
  </xs:sequence>  
</xs:complexType>  
  
</xs:element>*

*The Schema above is interpreted like this:*

* *<xs:element name="note"> defines the element called "note"*
* *<xs:complexType> the "note" element is a complex type*
* *<xs:sequence> the complex type is a sequence of elements*
* *<xs:element name="to" type="xs:string"> the element "to" is of type string (text)*
* *<xs:element name="from" type="xs:string"> the element "from" is of type string*
* *<xs:element name="heading" type="xs:string"> the element "heading" is of type string*
* *<xs:element name="body" type="xs:string"> the element "body" is of type string*

*XML Schemas are More Powerful than DTD*

* *XML Schemas are written in XML*
* *XML Schemas are extensible to additions*
* *XML Schemas support data types*
* *XML Schemas support namespaces*

*Why Use an XML Schema?*

*With XML Schema, your XML files can carry a description of its own format.*

*With XML Schema, independent groups of people can agree on a standard for interchanging data.*

*With XML Schema, you can verify data.*

*XML Schemas Support Data Types*

*One of the greatest strength of XML Schemas is the support for data types:*

* *It is easier to describe document content*
* *It is easier to define restrictions on data*
* *It is easier to validate the correctness of data*
* *It is easier to convert data between different data types*

*XML Schemas use XML Syntax*

*Another great strength about XML Schemas is that they are written in XML:*

* *You don't have to learn a new language*
* *You can use your XML editor to edit your Schema files*
* *You can use your XML parser to parse your Schema files*
* *You can manipulate your Schemas with the XML DOM*
* *You can transform your Schemas with XSLT*

*If you want to study XML Schema, please read our*[*XML Schema Tutorial*](https://www.w3schools.com/xml/schema_intro.asp)*.*

*// значыць, лепей выкарыстоўваць менавіта схемы!!! //*

[*https://www.w3schools.com/xml/xml\_server.asp*](https://www.w3schools.com/xml/xml_server.asp)

*XML on the Server*

*XML files are plain text files just like HTML files.*

*XML can easily be stored and generated by a standard web server.*

*Storing XML Files on the Server*

*XML files can be stored on an Internet server exactly the same way as HTML files.*

*Start Windows Notepad and write the following lines:*

*<?xml version="1.0" encoding="UTF-8"?>  
<note>  
  <from>Jani</from>  
  <to>Tove</to>  
  <message>Remember me this weekend</message>  
</note>*

*Save the file on your web server with a proper name like "note.xml".*

*Generating XML with PHP*

*XML can be generated on a server without any installed XML software.*

*To generate an XML response from the server using PHP, use following code:*

*<?php  
header("Content-type: text/xml");  
echo "<?xml version='1.0' encoding='UTF-8****'****?>";  
echo "<note>";  
echo "<from>Jani</from>";  
echo "<to>Tove</to>";  
echo "<message>Remember me this weekend</message>";  
echo "</note>";  
?>*

*Note that the content type of the response header must be set to "text/xml".*

[*See how the PHP file will be returned from the server*](https://www.w3schools.com/xml/note.php)*.*

*If you want to study PHP, you will find our PHP tutorial on our*[*homepage*](https://www.w3schools.com/default.asp)*.*

*Generating XML with ASP*

*To generate an XML response from the server - simply write the following code and save it as an ASP file on the web server:*

*<%  
response.ContentType="text/xml"  
response.Write("<?xml version='1.0' encoding='UTF-8****'****?>")  
response.Write("<note>")  
response.Write("<from>Jani</from>")  
response.Write("<to>Tove</to>")  
response.Write("<message>Remember me this weekend</message>")  
response.Write("</note>")  
%>*

*Note that the content type of the response must be set to "text/xml".*

[*See how the ASP file will be returned from the server*](https://www.w3schools.com/xml/note.asp)*.*

*If you want to study ASP, you will find our ASP tutorial on our*[*homepage*](https://www.w3schools.com/default.asp)*.*

*Generating XML From a Database*

*XML can be generated from a database without any installed XML software.*

*To generate an XML database response from the server, simply write the following code and save it as an ASP file on the web server:*

*<%  
response.ContentType = "text/xml"  
set conn=Server.CreateObject("ADODB.Connection")  
conn.provider="Microsoft.Jet.OLEDB.4.0;"  
conn.open server.mappath("/datafolder/database.mdb")  
  
sql="select fname,lname from tblGuestBook"  
set rs=Conn.Execute(sql)  
  
response.write("<?xml version='1.0' encoding='UTF-8****'****?>")  
response.write("<guestbook>")  
while (not rs.EOF)  
response.write("<guest>")  
response.write("<fname>" & rs("fname") & "</fname>")  
response.write("<lname>" & rs("lname") & "</lname>")  
response.write("</guest>")  
rs.MoveNext()  
wend  
  
rs.close()  
conn.close()  
response.write("</guestbook>")  
%>*

[*See the real life database output from the ASP file above*](https://www.w3schools.com/xml/guestbook.asp)*.*

*The example above uses ASP with ADO.*

*If you want to study ASP and ADO, you will find the tutorials on our*[*homepage*](https://www.w3schools.com/default.asp)*.*

*// intbu than this example //*

*Transforming XML with XSLT on the Server*

*This ASP transforms an XML file to XHTML on the server:*

*<%  
'Load XML  
set xml = Server.CreateObject("Microsoft.XMLDOM")  
xml.async = false  
xml.load(Server.MapPath("simple.xml"))  
  
'Load XSL  
set xsl = Server.CreateObject("Microsoft.XMLDOM")  
xsl.async = false  
xsl.load(Server.MapPath("simple.xsl"))  
  
'Transform file  
Response.Write(xml.transformNode(xsl))  
%>*

*Example explained*

* *The first block of code creates an instance of the Microsoft XML parser (XMLDOM), and loads the XML file into memory.*
* *The second block of code creates another instance of the parser and loads the XSL file into memory.*
* *The last line of code transforms the XML document using the XSL document, and sends the result as XHTML to your browser. Nice!*

[*https://www.w3schools.com/xml/xml\_examples.asp*](https://www.w3schools.com/xml/xml_examples.asp)

*XML Examples*

…

*// прапускаем, бо, здаецца, мне ўсё ясна – і я трошку спяшаюся. //*

[*https://www.w3schools.com/xml/xml\_quiz.asp*](https://www.w3schools.com/xml/xml_quiz.asp)

*XML Quiz*

*You can test your XML skills with W3Schools' Quiz.*

*The Test*

*The test contains 25 questions and there is no time limit. The test is not official, it's just a nice way to see how much you know, or don't know, about XML. // важная крыніца навучання //*

*Count Your Score*

*You will get 1 point for each correct answer. At the end of the Quiz, your total score will be displayed. Maximum score is 25 points.*

*Good luck!*[*Start the XML Quiz*](https://www.w3schools.com/quiztest/quiztest.asp?qtest=XML)

*[](https://www.w3schools.com/cert/default.asp)*

*W3Schools' Online Certification*

*...*

*// у тэсце адказаў на 21 з 25 пытанняў //*

[*https://www.w3schools.com/xml/xml\_exam.asp*](https://www.w3schools.com/xml/xml_exam.asp)

*W3Schools XML Certificate*

|  |  |  |
| --- | --- | --- |
| *W3Schools Certified* |  | *W3Schools offers an Online Certification Program.*  *The perfect solution for busy professionals who need to balance work, family, and career building.*  *More than 15 000 certificates already issued!* |

|  |  |
| --- | --- |
| *Document Your Skills*  *Knowledge is power, especially in the current job market. Documentation of your skills enables you to advance your career, or help you to start a new one.*  *Getting a certificate proves your commitment to upgrade your skills, gives you the credibility needed for more responsibilities, larger projects, and a higher salary.* | *W3Schools Certificate* |

*How Does It Work?*

* *Study for free at W3Schools.com*
* *Study at your own speed*
* *Test your skills with W3Schools online quizzes*
* *Apply for your certificate by paying an exam fee*
* *Take your exam online, at any time, and from any location*

AJAX

[*https://www.w3schools.com/xml/ajax\_intro.asp*](https://www.w3schools.com/xml/ajax_intro.asp)

*AJAX Introduction*

*AJAX is a developer's 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*

*Try it Yourself Examples in Every Chapter*

*In every chapter, you can edit the examples online, and click on a button to view the result.*

*AJAX Example*

*…*

*AJAX Example Explained*

*…*

*What is AJAX?*

*AJAX =****A****synchronous****J****avaScript****A****nd****X****ML.*

*AJAX is not a programming language.*

*AJAX just uses a combination of:*

* *A browser built-in XMLHttpRequest object (to request data from a web server)*
* *JavaScript and HTML DOM (to display or use the data)*

*AJAX is a misleading name. AJAX applications might use XML to transport data, but it is equally common to transport data as plain text or JSON text.*

*AJAX allows web pages to be updated asynchronously by exchanging data with a web server behind the scenes. This means that it is possible to update parts of a web page, without reloading the whole page.*

*How AJAX Works*

**

* *1. An event occurs in a web page (the page is loaded, a button is clicked)*
* *2. An XMLHttpRequest object is created by JavaScript*
* *3. The XMLHttpRequest object sends a request to a web server*
* *4. The server processes the request*
* *5. The server sends a response back to the web page*
* *6. The response is read by JavaScript*
* *7. Proper action (like page update) is performed by JavaScript*

[*https://www.w3schools.com/xml/ajax\_xmlhttprequest\_create.asp*](https://www.w3schools.com/xml/ajax_xmlhttprequest_create.asp)

*AJAX - The XMLHttpRequest Object*

*The keystone of AJAX is the XMLHttpRequest object.*

*The XMLHttpRequest Object*

*All modern browsers support the XMLHttpRequest object. The XMLHttpRequest object is used to exchange data with a server behind the scenes. This means that it is possible to update parts of a web page, without reloading the whole page.*

*Create an XMLHttpRequest Object*

*All modern browsers (Chrome, IE7+, Firefox, Safari, and Opera) have a built-in XMLHttpRequest object.*

*Syntax for creating an XMLHttpRequest object:*

*variable = new XMLHttpRequest();*

*Old versions of Internet Explorer (IE5 and IE6) use an ActiveX Object:*

*…*

*To handle all browsers, including IE5 and IE6, check if the browser supports the XMLHttpRequest object. If it does, create an XMLHttpRequest object, if not, create an ActiveXObject:*

*Example*

*var xhttp;  
if (window.XMLHttpRequest) {  
    xhttp = new XMLHttpRequest();  
    } else {  
    // code for IE6, IE5  
    xhttp = new ActiveXObject("Microsoft.XMLHTTP");  
}*

*Access Across Domains*

*For security reasons, modern browsers do not allow access across domains. This means that both the web page and the XML file it tries to load, must be located on the same server. // моцнае абмежаванне.* // *Чаму так бяспечней? Што было інакш? //*

*The examples on W3Schools all open XML files located on the W3Schools domain. If you want to use the example above on one of your own web pages, the XML files you load must be located on your own server.*

*Old Versions of Internet Explorer (IE5 and IE6)*

*…*

*XMLHttpRequest Object Methods*

|  |  |
| --- | --- |
| ***Method*** | ***Description*** |
| *new XMLHttpRequest()* | *Creates a new XMLHttpRequest object* |
| *abort()* | *Cancels the current request* |
| *getAllResponseHeaders()* | *Returns header information* |
| *getResponseHeader()* | *Returns specific header information* |
| *open(method,url,async,user,psw)* | *Specifies the request  method: the request type GET or POST url: the file location async: true (asynchronous) or false (synchronous) user: optional user name psw: optional password* |
| *send()* | *Sends the request to the server Used for GET requests* |
| *send(string)* | *Sends the request to the server. Used for POST requests* |
| *setRequestHeader()* | *Adds a label/value pair to the header to be sent* |

*XMLHttpRequest Object Properties*

|  |  |
| --- | --- |
| ***Property*** | ***Description*** |
| *onreadystatechange* | *Defines a function to be called when the readyState property changes* |
| *readyState* | *Holds the status of the XMLHttpRequest. 0: request not initialized  1: server connection established 2: request received  3: processing request  4: request finished and response is ready* |
| *responseText* | *Returns the response data as a string* |
| *responseXML* | *Returns the response data as XML data* |
| *status* | *Returns the status-number of a request 200: "OK" 403: "Forbidden" 404: "Not Found" For a complete list go to the*[*Http Messages Reference*](https://www.w3schools.com/tags/ref_httpmessages.asp) |
| *statusText* | *Returns the status-text (e.g. "OK" or "Not Found")* |

[*https://www.w3schools.com/xml/ajax\_xmlhttprequest\_send.asp*](https://www.w3schools.com/xml/ajax_xmlhttprequest_send.asp)

*AJAX - Send a Request To a Server*

*The XMLHttpRequest object is used to exchange data with a server.*

*Send a Request To a Server*

*To send a request to a server, we use the open() and send() methods of the XMLHttpRequest object:*

*xhttp.open("GET", "ajax\_info.txt", true);  
xhttp.send();*

|  |  |
| --- | --- |
| ***Method*** | ***Description*** |
| *open(method, url, async)* | *Specifies the type of request  method: the type of request: GET or POST url: the server (file) location async: true (asynchronous) or false (synchronous)* |
| *send()* | *Sends the request to the server (used for GET)* |
| *send(string)* | *Sends the request to the server (used for POST)* |

*GET or POST?*

*GET is simpler and faster than POST, and can be used in most cases.*

*However, always use POST requests when:*

* *A cached file is not an option (update a file or database on the server).*
* *Sending a large amount of data to the server (POST has no size limitations).*
* *Sending user input (which can contain unknown characters), POST is more robust and secure than GET.*

*GET Requests*

*A simple GET request:*

*Example*

*xhttp.open("GET", "demo\_get.asp", true);  
xhttp.send();*

*In the example above, you may get a cached result. To avoid this, add a unique ID to the URL:*

*Example*

*xhttp.open("GET", "demo\_get.asp?t=" + Math.random(), true);  
xhttp.send();*

*If you want to send information with the GET method, add the information to the URL:*

*Example*

*xhttp.open("GET", "demo\_get2.asp?fname=Henry&lname=Ford", true);  
xhttp.send();*

*POST Requests*

*A simple POST request:*

*Example*

*xhttp.open("POST", "demo\_post.asp", true);  
xhttp.send();*

*To POST data like an HTML form, add an HTTP header with setRequestHeader(). Specify the data you want to send in the send() method:*

*Example*

*xhttp.open("POST", "ajax\_test.asp", true);  
xhttp.setRequestHeader("Content-type", "application/x-www-form-urlencoded");  
xhttp.send("fname=Henry&lname=Ford");*

|  |  |
| --- | --- |
| ***Method*** | ***Description*** |
| *setRequestHeader(header, value)* | *Adds HTTP headers to the request  header: specifies the header name value: specifies the header value* |

*The url - A File On a Server*

*The url parameter of the open() method, is an address to a file on a server:*

*xhttp.open("GET", "ajax\_test.asp", true);*

*The file can be any kind of file, like .txt and .xml, or server scripting files like .asp and .php (which can perform actions on the server before sending the response back).*

*Asynchronous - True or False?*

*To send the request asynchronously, the async parameter of the open() method has to be set to true:*

*xhttp.open("GET", "ajax\_test.asp", true);*

*Sending asynchronous requests is a huge improvement for web developers. Many of the tasks performed on the server are very time consuming. Before AJAX, this operation could cause the application to hang or stop.*

*By sending asynchronously, the JavaScript does not have to wait for the server response, but can instead:*

* *execute other scripts while waiting for server response*
* *deal with the response when the response is ready*

*Async = true*

*When using async = true, specify a function to execute when the response is ready in the onreadystatechange event:*

*Example*

*xhttp.onreadystatechange = function() {  
  if (this.readyState == 4 && this.status == 200) {  
    document.getElementById("demo").innerHTML = this.responseText;  
  }  
};  
xhttp.open("GET", "ajax\_info.txt", true);  
xhttp.send();*

*You will learn more about onreadystatechange in a later chapter.*

*Async = false*

*To use async=false, change the third parameter in the open() method to false:*

*xhttp.open("GET", "ajax\_info.txt", false);*

*Using async=false is not recommended, but for a few small requests this can be ok.*

*Remember that the JavaScript will NOT continue to execute, until the server response is ready. If the server is busy or slow, the application will hang or stop.*

***Note:****When you use async=false, do NOT write an onreadystatechange function - just put the code after the send() statement: // intbu after this warning //*

*Example*

*xhttp.open("GET", "ajax\_info.txt", false);  
xhttp.send();  
document.getElementById("demo").innerHTML = xhttp.responseText;*

[*https://www.w3schools.com/xml/ajax\_xmlhttprequest\_response.asp*](https://www.w3schools.com/xml/ajax_xmlhttprequest_response.asp)

*AJAX - Server Response*

*The onreadystatechange Property*

*The****readyState****property holds the status of the XMLHttpRequest.*

*The****onreadystatechange****property defines a function to be executed when the readyState changes.*

*The****status****property and the****statusText****property holds the status of the XMLHttpRequest object.*

|  |  |
| --- | --- |
| ***Property*** | ***Description*** |
| *onreadystatechange* | *Defines a function to be called when the readyState property changes* |
| *readyState* | *Holds the status of the XMLHttpRequest.  0: request not initialized  1: server connection established 2: request received  3: processing request  4: request finished and response is ready* |
| *status* | *200: "OK" 403: "Forbidden" 404: "Page not found" For a complete list go to the*[*Http Messages Reference*](https://www.w3schools.com/tags/ref_httpmessages.asp) |
| *statusText* | *Returns the status-text (e.g. "OK" or "Not Found")* |

*The onreadystatechange function is called every time the readyState changes. When readyState is 4 and status is 200, the response is ready:*

*Example*

*function loadDoc() {  
    var xhttp = new XMLHttpRequest();  
    xhttp.onreadystatechange = function() {  
        if (this.readyState == 4 && this.status == 200) {  
            document.getElementById("demo").innerHTML =  
            this.responseText;  
       }  
    };  
    xhttp.open("GET", "ajax\_info.txt", true);  
    xhttp.send();   
}*

*The onreadystatechange event is triggered four times (1-4), one time for each change in the readyState.*

*Using a Callback Function*

*A callback function is a function passed as a parameter to another function.*

*If you have more than one AJAX task in a website, you should create one function for executing the XMLHttpRequest object, and one callback function for each AJAX task.*

*The function call should contain the URL and what function to call when the response is ready.*

*Example*

*loadDoc("url-1", myFunction1);  
  
loadDoc("url-2", myFunction2);  
  
function loadDoc(url, cFunction) {  
  var xhttp;  
  xhttp=new XMLHttpRequest();  
  xhttp.onreadystatechange = function() {  
    if (this.readyState == 4 && this.status == 200) {  
      cFunction(this);  
    }  
 };  
  xhttp.open("GET", url, true);  
  xhttp.send();  
}  
  
function myFunction1(xhttp) {  
  // action goes here  
}   
function myFunction2(xhttp) {  
  // action goes here  
}*

*Server Response Properties*

|  |  |
| --- | --- |
| ***Property*** | ***Description*** |
| *responseText* | *get the response data as a string* |
| *responseXML* | *get the response data as XML data* |

*Server Response Methods*

|  |  |
| --- | --- |
| ***Method*** | ***Description*** |
| *getResponseHeader()* | *Returns specific header information from the server resource* |
| *getAllResponseHeaders()* | *Returns all the header information from the server resource* |

*The responseText Property*

*The****responseText****property returns the server response as a JavaScript string, and you can use it accordingly:*

*Example*

*document.getElementById("demo").innerHTML = xhttp.responseText;*

*The responseXML Property*

*The XML HttpRequest object has an in-built XML parser.*

*The****responseXML****property returns the server response as an XML DOM object.*

*Using this property you can parse the response as an XML DOM object:*

*Example*

*Request the file*[*cd\_catalog.xml*](https://www.w3schools.com/xml/cd_catalog.xml)*and parse the response:*

*xmlDoc = xhttp.responseXML;  
txt = "";  
x = xmlDoc.getElementsByTagName("ARTIST");  
for (i = 0; i < x.length; i++) {  
  txt += x[i].childNodes[0].nodeValue + "<br>";  
  }  
document.getElementById("demo").innerHTML = txt;  
xhttp.open("GET", "cd\_catalog.xml", true);  
xhttp.send();*

[*//*](https://www.w3schools.com/xml/tryit.asp?filename=tryajax_responsexml) *не магу сказаць, што апошнія радкі гэтага прыклада мне ясны //*

*You will learn a lot more about XML DOM in the DOM chapters of this tutorial.*

*The getAllResponseHeaders() Method*

*The****getAllResponseHeaders()****method returns all header information from the server response.*

*Example*

*var xhttp = new XMLHttpRequest();  
xhttp.onreadystatechange = function() {  
  if (this.readyState == 4 && this.status == 200) {  
    document.getElementById("demo").innerHTML =  
    this.getAllResponseHeaders();  
  }  
};*

*// з прыклада бачна, што ўваходзіць у хедэр //*

*The getResponseHeader() Method*

*The****getResponseHeader()****method returns specific header information from the server response.*

*Example*

*var xhttp = new XMLHttpRequest();  
xhttp.onreadystatechange = function() {  
  if (this.readyState == 4 && this.status == 200) {  
    document.getElementById("demo").innerHTML =  
    this.getResponseHeader("Last-Modified");  
  }  
};  
xhttp.open("GET", "ajax\_info.txt", true);  
xhttp.send();*

[*//*](https://www.w3schools.com/xml/tryit.asp?filename=tryajax_lastmodified) *метад вяртае пэўны спецыфічны хедэр. Яны, відаць, вызначаны ў пратаколах //*

[*https://www.w3schools.com/xml/ajax\_xmlfile.asp*](https://www.w3schools.com/xml/ajax_xmlfile.asp)

*AJAX XML Example*

*AJAX can be used for interactive communication with an XML file.*

*AJAX XML Example*

*The following example will demonstrate how a web page can fetch information from an XML file with AJAX:*

*…*

*Example Explained*

*When a user clicks on the "Get CD info" button above, the loadDoc() function is executed.*

*The loadDoc() function creates an XMLHttpRequest object, adds the function to be executed when the server response is ready, and sends the request off to the server.*

*When the server response is ready, an HTML table is built, nodes (elements) are extracted from the XML file, and it finally updates the element "demo" with the HTML table filled with XML data:*

*…*

*// прыклад, у цэлым, мне ясны //*

[*https://www.w3schools.com/xml/ajax\_php.asp*](https://www.w3schools.com/xml/ajax_php.asp)

*AJAX PHP Example*

*AJAX is used to create more interactive applications.*

*AJAX PHP Example*

*The following example demonstrates how a web page can communicate with a web server while a user types characters in an input field:*

*Example Explained*

*In the example above, when a user types a character in the input field, a function called "showHint()" is executed.*

*The function is triggered by the onkeyup event.*

*Here is the HTML code:*

*…*

*// прыклад мне ясны за тым выключэннем, што я не зразумеў фармат запыта да html-старонкі //*

*Code explanation:*

*First, check if the input field is empty (str.length == 0). If it is, clear the content of the txtHint placeholder and exit the function.*

*However, if the input field is not empty, do the following:*

* *Create an XMLHttpRequest object*
* *Create the function to be executed when the server response is ready*
* *Send the request off to a PHP file (gethint.php) on the server*
* *Notice that q parameter is added gethint.php?q="+str*
* *The str variable holds the content of the input field*

*// гэта, відаць, спосаб даслаць php-файлу параметр //*

## The PHP File - "gethint.php"

The PHP file checks an array of names, and returns the corresponding name(s) to the browser:

<?php  
// Array with names  
$a[] = "Anna";  
$a[] = "Brittany";  
$a[] = "Cinderella";  
$a[] = "Diana";  
$a[] = "Eva";  
$a[] = "Fiona";  
$a[] = "Gunda";  
$a[] = "Hege";  
$a[] = "Inga";  
$a[] = "Johanna";  
$a[] = "Kitty";  
$a[] = "Linda";  
$a[] = "Nina";  
$a[] = "Ophelia";  
$a[] = "Petunia";  
$a[] = "Amanda";  
$a[] = "Raquel";  
$a[] = "Cindy";  
$a[] = "Doris";  
$a[] = "Eve";  
$a[] = "Evita";  
$a[] = "Sunniva";  
$a[] = "Tove";  
$a[] = "Unni";  
$a[] = "Violet";  
$a[] = "Liza";  
$a[] = "Elizabeth";  
$a[] = "Ellen";  
$a[] = "Wenche";  
$a[] = "Vicky";  
  
// get the q parameter from URL  
$q = $\_REQUEST["q"];  
  
$hint = "";  
  
// lookup all hints from array if $q is different from ""   
if ($q !== "") {  
    $q = strtolower($q);  
    $len=strlen($q);  
    foreach($a as $name) {  
        if (stristr($q, substr($name, 0, $len))) {  
            if ($hint === "") {  
                $hint = $name;  
            } else {  
                $hint .= ", $name";  
            }  
        }  
    }  
}  
  
// Output "no suggestion" if no hint was found or output correct values   
echo $hint === "" ? "no suggestion" : $hint;  
?>

*// код, з большага, зразумеў. Відаць, echo азначае адказ назад кліенту //*

[*https://www.w3schools.com/xml/ajax\_asp.asp*](https://www.w3schools.com/xml/ajax_asp.asp)

*AJAX ASP Example*

*AJAX is used to create more interactive applications.*

*AJAX ASP Example*

*The following example will demonstrate how a web page can communicate with a web server while a user type characters in an input field:*

*Example Explained*

*In the example above, when a user types a character in the input field, a function called "showHint()" is executed.*

*The function is triggered by the onkeyup event.*

*Here is the HTML code:*

*// тлумачэнне ідэнтычна мінуламу. Фармат запыту да файла .asp падобны на .php //*

*Code explanation:*

*First, check if the input field is empty (str.length == 0). If it is, clear the content of the txtHint placeholder and exit the function.*

*However, if the input field is not empty, do the following:*

* *Create an XMLHttpRequest object*
* *Create the function to be executed when the server response is ready*
* *Send the request off to an ASP file (gethint.asp) on the server*
* *Notice that q parameter is added gethint.asp?q="+str*
* *The str variable holds the content of the input field*

## The ASP File - "gethint.asp"

The ASP file checks an array of names, and returns the corresponding name(s) to the browser:

<%  
response.expires=-1  
dim a(30)  
'Fill up array with names  
a(1)="Anna"  
a(2)="Brittany"  
a(3)="Cinderella"  
a(4)="Diana"  
a(5)="Eva"  
a(6)="Fiona"  
a(7)="Gunda"  
a(8)="Hege"  
a(9)="Inga"  
a(10)="Johanna"  
a(11)="Kitty"  
a(12)="Linda"  
a(13)="Nina"  
a(14)="Ophelia"  
a(15)="Petunia"  
a(16)="Amanda"  
a(17)="Raquel"  
a(18)="Cindy"  
a(19)="Doris"  
a(20)="Eve"  
a(21)="Evita"  
a(22)="Sunniva"  
a(23)="Tove"  
a(24)="Unni"  
a(25)="Violet"  
a(26)="Liza"  
a(27)="Elizabeth"  
a(28)="Ellen"  
a(29)="Wenche"  
a(30)="Vicky"  
  
'get the q parameter from URL  
q=ucase(request.querystring("q"))  
  
'lookup all hints from array if length of q>0  
if len(q)>0 then  
  hint=""  
  for i=1 to 30  
    if q=ucase(mid(a(i),1,len(q))) then  
      if hint="" then  
        hint=a(i)  
      else  
        hint=hint & " , " & a(i)  
      end if  
    end if  
  next  
end if  
  
'Output "no suggestion" if no hint were found  
'or output the correct values  
if hint="" then  
  response.write("no suggestion")  
else  
  response.write(hint)  
end if  
%>

*// трошку адрозніваецца сінтаксіс. Але, у цэлым, з большага, усё ясна. Я так разумею, у дадзеных прыкладах кліент і сервер змешчаны на адным і тым жа серверы? Я паспрабаваў. Не атрымалася. Не работае. Старонка адлюстроўваецца, але сувязі з php файлам няма //*

[*https://www.w3schools.com/xml/ajax\_database.asp*](https://www.w3schools.com/xml/ajax_database.asp)

*AJAX Database Example*

*// спачатку ўсё аналагічна, як і ў мінулым параграфе. Выкарыстоўваюць файл .asp //*

*The AJAX Server Page*

*The page on the server called by the JavaScript above is an ASP file called "getcustomer.asp".*

*The server file could easily be rewritten in PHP, or some other server languages. // л.з.! перавагі і недахопы тых ці іншых серверных моваў… //*

[*Look at a corresponding example in PHP*](https://www.w3schools.com/php/php_ajax_database.asp)*.*

*The source code in "getcustomer.asp" runs a query against a database, and returns the result in an HTML table:*

<%  
response.expires=-1  
sql="SELECT \* FROM CUSTOMERS WHERE CUSTOMERID="  
sql=sql & "'" & request.querystring("q") & "'"  
  
set conn=Server.CreateObject("ADODB.Connection")  
conn.Provider="Microsoft.Jet.OLEDB.4.0"  
conn.Open(Server.Mappath("/datafolder/northwind.mdb"))  
set rs=Server.CreateObject("ADODB.recordset")  
rs.Open sql,conn  
  
response.write("<table>")  
do until rs.EOF  
  for each x in rs.Fields  
    response.write("<tr><td><b>" & x.name & "</b></td>")  
    response.write("<td>" & x.value & "</td></tr>")  
  next  
  rs.MoveNext  
loop  
response.write("</table>")  
%>

*// запыт да базы дадзен я не зразумеў. Але ў цэлым ясна. Адказ быў вернуты ў выглядзе тэкставага радку //*

[*https://www.w3schools.com/xml/ajax\_applications.asp*](https://www.w3schools.com/xml/ajax_applications.asp)

*XML Applications*

*This chapter demonstrates some HTML applications using XML, HTTP, DOM, and JavaScript.*

*The XML Document Used*

*In this chapter we will use the XML file called*[*"cd\_catalog.xml"*](https://www.w3schools.com/xml/cd_catalog.xml)*.*

*Display XML Data in an HTML Table*

*This example loops through each <CD> element, and displays the values of the <ARTIST> and the <TITLE> elements in an HTML table:*

*…*

*For more information about using JavaScript and the XML DOM, go to*[*DOM Intro.*](https://www.w3schools.com/xml/dom_intro.asp)

*// прыведзены прыклад мне, у цэлым, ясны //*

*Display the First CD in an HTML div Element*

*This example uses a function to display the first CD element in an HTML element with id="showCD":*

*…*

*// прыклад таксама ясны. Наступны прыклад таксама ясны – там можна ажыццяўляць навігацыю. У апошнім прыкладзе цудоўна паказана, што з кожным візуальным элементам html-старонкі можна злучыць скрыпт. І гэта вельмі-вельмі цудоўна для стварэння інтэрфэйсаў //*

[*https://www.w3schools.com/xml/ajax\_examples.asp*](https://www.w3schools.com/xml/ajax_examples.asp)

*// тут пералічаны ўсе прыклады //*

DOM

[*https://www.w3schools.com/xml/dom\_intro.asp*](https://www.w3schools.com/xml/dom_intro.asp)

*// пачынаем вывучаць DOM //*

*XML DOM Tutorial*

*XML DOM*

**

*What is the DOM?*

*The DOM defines a standard for accessing and manipulating documents:*

*"The W3C Document Object Model (DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document."*

*The HTML DOM defines a standard way for accessing and manipulating HTML documents. It presents an HTML document as a tree-structure.*

*The XML DOM defines a standard way for accessing and manipulating XML documents. It presents an XML document as a tree-structure.*

*Understanding the DOM is a must for anyone working with HTML or XML.*

*The HTML DOM*

*All HTML elements can be accessed through the HTML DOM.*

*This example changes the value of an HTML element with id="demo":*

*Example*

*…*

*This example changes the value of the first <h1> element in an HTML document:*

*Example*

*…*

*// прыклады для мяне зразумелыя //*

***Note:****Even if the HTML document contains only ONE <h1> element you still have to specify the array index [0], because the getElementsByTagName() method always returns an array.*

*You can learn a lot more about the HTML DOM in our*[*JavaScript tutorial*](https://www.w3schools.com/js/default.asp)*.*

*The XML DOM*

*All XML elements can be accessed through the XML DOM.*

*The XML DOM is:*

* *A standard object model for XML*
* *A standard programming interface for XML*
* *Platform- and language-independent*
* *A W3C standard*

*In other words:****The XML DOM is a standard for how to get, change, add, or delete XML elements.***

*Get the Value of an XML Element*

*This code retrieves the text value of the first <title> element in an XML document:*

*Example*

*txt = xmlDoc.getElementsByTagName("title")[0].childNodes[0].nodeValue;*

*Loading an XML File*

*The XML file used in the examples below is*[*books.xml*](javascript:void(0))*.*

*This example reads "books.xml" into xmlDoc and retrieves the text value of the first <title> element in books.xml:*

*…*

*// прыклад для мяне ясны //*

*Loading an XML String*

*This example loads a text string into an XML DOM object, and extracts the info from it with JavaScript:*

*Example*

*<html>  
<body>  
  
<p id="demo"></p>  
  
<script>  
var text, parser, xmlDoc;  
  
text = "<bookstore><book>" +  
"<title>Everyday Italian</title>" +  
"<author>Giada De Laurentiis</author>" +  
"<year>2005</year>" +  
"</book></bookstore>";  
  
parser = new DOMParser();  
xmlDoc = parser.parseFromString(text,"text/xml");  
  
document.getElementById("demo").innerHTML =****xmlDoc.getElementsByTagName("title")[0].childNodes[0].nodeValue****;  
</script>  
  
</body>  
</html>*

*Programming Interface*

*The DOM models XML as a set of node objects. The nodes can be accessed with JavaScript or other programming languages. In this tutorial we use JavaScript.*

*The programming interface to the DOM is defined by a set of standard properties and methods.*

***Properties****are often referred to as something that is (i.e. nodename is "book").*

***Methods****are often referred to as something that is done (i.e. delete "book").*

*XML DOM Properties*

*These are some typical DOM properties:*

* *x.nodeName - the name of x*
* *x.nodeValue - the value of x*
* *x.parentNode - the parent node of x*
* *x.childNodes - the child nodes of x*
* *x.attributes - the attributes nodes of x*

*Note: In the list above, x is a node object.*

*XML DOM Methods*

* *x.getElementsByTagName(name) - get all elements with a specified tag name*
* *x.appendChild(node) - insert a child node to x*
* *x.removeChild(node) - remove a child node from x*

*Note: In the list above, x is a node object.*

[*https://www.w3schools.com/xml/dom\_nodes.asp*](https://www.w3schools.com/xml/dom_nodes.asp)

*XML 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*

*DOM Example*

*Look at the following XML file (*[*books.xml*](javascript:void(0))*):*

*…*

*// прыклад я зразумеў //*

*Text is Always Stored in Text Nodes*

*A common error in DOM processing is to expect an element node to contain text.*

*However, the text of an element node is stored in a text node.*

*In this example:****<year>2005</year>****, the element node <year> holds a text node with the value "2005".*

*"2005" is****not****the value of the <year> element!*

*The XML DOM Node Tree*

*The XML DOM views an XML document as a tree-structure. The tree structure is 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.*

*The node tree shows the set of nodes, and the connections between them. The tree starts at the root node and branches out to the text nodes at the lowest level of the tree: // адметная рыса дадзенага дрэва //*

**

*The image above represents the XML file*[*books.xml*](javascript:void(0))*.*

*Node Parents, Children, and Siblings*

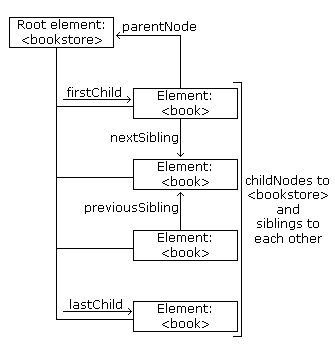
*The nodes in the node tree have a hierarchical relationship to each other.*

*// тут бачна, што калі апусціць некаторыя фрагменты тэксту і злучыць іншыя, то сэнс можа не змяніцца. Даследваць дадзеную асаблівасць тэкста. //*

*The terms parent, child, and sibling are used to describe the relationships. Parent nodes have children. Children on the same level are called siblings (brothers or sisters).*

* *In a node tree, the top node is called the root*
* *Every node, except the root, has exactly one parent node*
* *A node can have any number of children*
* *A leaf is a node with no children*
* *Siblings are nodes with the same parent*

*The following image illustrates a part of the node tree and the relationship between the nodes:*

**

*Because the XML data is structured in a tree form, it can be traversed without knowing the exact structure of the tree and without knowing the type of data contained within. // вялікая перавага дрэвавідных структур. Паразважаць. //*

*You will learn more about traversing the node tree in a later chapter of this tutorial.*

*First Child - Last Child*

*… // I understood this example //*

[*https://www.w3schools.com/xml/dom\_nodes\_access.asp*](https://www.w3schools.com/xml/dom_nodes_access.asp)

*XML DOM - Accessing Nodes*

*With the DOM, you can access every node in an XML document.*

*The examples below use the XML file*[*books.xml*](javascript:void(0))*.*

[*Access a node using its index number in a node list*](https://www.w3schools.com/xml/tryit.asp?filename=try_dom_index) *Use the getElementsByTagName() method to get the third <title> element in "books.xml"*

[*Loop through nodes using the length property*](https://www.w3schools.com/xml/tryit.asp?filename=try_dom_list_loop) *Use the length property to loop through all <title> elements in "books.xml"*

[*See the node type of an element*](https://www.w3schools.com/xml/tryit.asp?filename=try_dom_root) *Use the nodeType property to get node type of the root element in "books.xml".*

[*Loop through element nodes*](https://www.w3schools.com/xml/tryit.asp?filename=try_dom_loop) *Use the nodeType property to only process element nodes in "books.xml".*

[*Loop through element nodes using node relationships*](https://www.w3schools.com/xml/tryit.asp?filename=try_dom_navigate) *Use the nodeType property and the nextSibling property to process element nodes in "books.xml".*

*Accessing Nodes*

*You can access a node in three ways:*

*1. By using the getElementsByTagName() method*

*2. By looping through (traversing) the nodes tree.*

*3. By navigating the node tree, using the node relationships.*

*The getElementsByTagName() Method*

*getElementsByTagName() returns all elements with a specified tag name.*

*Syntax*

*node.getElementsByTagName("tagname");*

*Example*

*The following example returns all <title> elements under the x element:*

*x.getElementsByTagName("title");*

*Note that the example above only returns <title> elements under the x node. To return all <title> elements in the XML document use:*

*xmlDoc.getElementsByTagName("title");*

*where xmlDoc is the document itself (document node).*

*DOM Node List*

*The getElementsByTagName() method returns a node list. A node list is an array of nodes.*

*x = xmlDoc.getElementsByTagName("title");*

*The <title> elements in x can be accessed by index number. To access the third <title> you can write::*

*y = x[2];*

***Note:****The index starts at 0.*

*You will learn more about node lists in a later chapter of this tutorial.*

*DOM Node List Length*

*The length property defines the length of a node list (the number of nodes).*

*You can loop through a node list by using the length property:*

*Example*

*var x = xmlDoc.getElementsByTagName("title");  
  
for (i = 0; i <x.length; i++) {  
  // do something for each node  
  }*

*Node Types*

*The****documentElement****property of the XML document is the root node.*

*The****nodeName****property of a node is the name of the node.*

*The****nodeType****property of a node is the type of the node.*

*You will learn more about the node properties in the next chapter of this tutorial.*

*Traversing Nodes*

*The following code loops through the child nodes, that are also element nodes, of the root node:*

*Example*

*txt = "";  
x = xmlDoc.documentElement.childNodes;  
  
for (i = 0; i <x.length; i++) {  
  // Process only element nodes (type 1)  
  if (x[i].nodeType == 1) {  
    txt += x[i].nodeName + "<br>";  
  }  
}*

*Example explained:*

1. *Suppose you have loaded "*[*books.xml*](javascript:void(0))*" into xmlDoc*
2. *Get the child nodes of the root element (xmlDoc)*
3. *For each child node, check the node type. If the node type is "1" it is an element node*
4. *Output the name of the node if it is an element node*

*Navigating Node Relationships*

*The following code navigates the node tree using the node relationships:*

*Example*

*x = xmlDoc.getElementsByTagName("book")[0];  
xlen = x.childNodes.length;  
y = x.firstChild;  
  
txt = "";  
for (i = 0; i <xlen; i++) {  
  // Process only element nodes (type 1)  
  if (y.nodeType == 1) {  
    txt += y.nodeName + "<br>";  
  }  
  y = y.nextSibling;  
}*

*Example explained:*

1. *Suppose you have loaded "*[*books.xml*](javascript:void(0))*"into xmlDoc*
2. *Get the child nodes of the first book element*
3. *Set the "y" variable to be the first child node of the first book element*
4. *For each child node (starting with the first child node "y"):*
5. *Check the node type. If the node type is "1" it is an element node*
6. *Output the name of the node if it is an element node*
7. *Set the "y" variable to be the next sibling node, and run through the loop again*

[*https://www.w3schools.com/xml/dom\_nodes\_info.asp*](https://www.w3schools.com/xml/dom_nodes_info.asp)

*XML DOM Node Information*

*The nodeName, nodeValue, and nodeType properties contain information about nodes.*

*…*

*Node Properties*

*In the XML DOM, each node is an****object****.*

*Objects have methods and properties, that can be accessed and manipulated by JavaScript.*

*Three important node properties are:*

* *nodeName*
* *nodeValue*
* *nodeType*

*The nodeName Property*

*The nodeName property specifies the name of a node.*

* *nodeName is read-only*
* *nodeName of an element node is the same as the tag name*
* *nodeName of an attribute node is the attribute name*
* *nodeName of a text node is always #text*
* *nodeName of the document node is always #document*

*The nodeValue Property*

*The nodeValue property specifies the value of a node.*

* *nodeValue for element nodes is undefined*
* *nodeValue for text nodes is the text itself*
* *nodeValue for attribute nodes is the attribute value*

*Get the Value of an Element*

*The following code retrieves the text node value of the first <title> element:*

*Example*

*…*

*Change the Value of an Element*

*…*

*The nodeType Property*

*The nodeType property specifies the type of node.*

*nodeType is read only.*

*The most important node types are:*

|  |  |
| --- | --- |
| ***Node type*** | ***NodeType*** |
| *Element* | *1* |
| *Attribute* | *2* |
| *Text* | *3* |
| *Comment* | *8* |
| *Document* | *9* |

*// а якія ёсць яшчэ? //*

[*https://www.w3schools.com/xml/dom\_nodes\_nodelist.asp*](https://www.w3schools.com/xml/dom_nodes_nodelist.asp)

*XML DOM Node List*

*A list of nodes is returned by the getElementsByTagName() method and the childNodes property.*

*…*

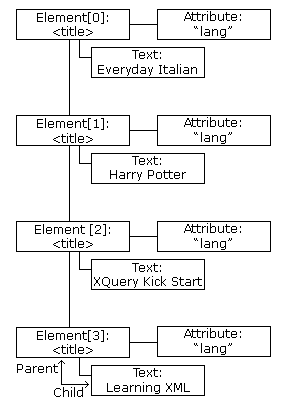
*DOM Node List*

*When using properties or methods like childNodes or getElementsByTagName(), a node list object is returned.*

*A node list object represents a list of nodes, in the same order as in the XML.*

*Nodes in the node list are accessed with index numbers starting from 0.*

*The following image represents a node list of the <title> elements in "*[*books.xml*](javascript:void(0))*":*

**

*Suppose "books.xml" is loaded into the variable xmlDoc.*

*This code fragment returns a node list of title elements in "books.xml":*

*x = xmlDoc.getElementsByTagName("title");*

*After the execution of the statement above, x is a node list object. // лепей зразумець, як працуюць скрыптавыя і веб-тэхналогіі фізічна, лагічна і праграмна //*

*The following code fragment returns the text from the first <title> element in the node list (x):*

*…*

*Node List Length*

*A node list object keeps itself up-to-date. If an element is deleted or added, the list is automatically updated.*

*The length property of a node list is the number of nodes in the list.*

*…*

*The length of the node list can be used to loop through all the elements in the list.*

*…*

*DOM Attribute List (Named Node Map)*

*The attributes property of an element node returns a list of attribute nodes.*

*This is called a named node map, and is similar to a node list, except for some differences in methods and properties.*

*An attribute list keeps itself up-to-date. If an attribute is deleted or added, the list is automatically updated.*

*…*

[*https://www.w3schools.com/xml/dom\_nodes\_traverse.asp*](https://www.w3schools.com/xml/dom_nodes_traverse.asp)

*XML DOM Traverse Node Tree*

*Traversing means looping through or traveling across the node tree.*

*Traversing the Node Tree*

*Often you want to loop an XML document, for example: when you want to extract the value of each element.*

*This is called "Traversing the node tree"*

*The example below loops through all child nodes of <book>, and displays their names and values:*

*…*

*// прыклад ясны акрамя наступных радкоў //*

*parser = new DOMParser();*

*xmlDoc = parser.parseFromString(text,"text/xml");*

*Browser Differences in DOM Parsing*

*All modern browsers support the W3C DOM specification.*

*However, there are some differences between browsers. One important difference is:*

* *The way they handle white-spaces and new lines*

*DOM - White Spaces and New Lines*

*XML often contains new line, or white space characters, between nodes. This is often the case when the document is edited by a simple editor like Notepad.*

*The following example (edited by Notepad) contains CR/LF (new line) between each line and two spaces in front of each child node:*

*…*

*Internet Explorer 9 and earlier do NOT treat empty white-spaces, or new lines as text nodes, while other browsers do. // вось праблемы з інтэрнэт-эксплорэрам, пра якія Саша Пасынкаў столь часта казаў //*

*The following example will output the number of child nodes the root element (of*[*books.xml*](https://www.w3schools.com/xml/books.xml)*) has. IE9 and earlier will output 4 child nodes, while IE10 and later versions, and other browsers will output 9 child nodes:*

*// гэты прыклад для мяне не такі ясны. Варта лепей зразумець //*

*PCDATA - Parsed Character Data*

*XML parsers normally parse all the text in an XML document.*

*When an XML element is parsed, the text between the XML tags is also parsed:*

*<message>This text is also parsed</message>*

*The parser does this because XML elements can contain other elements, as in this example, where the <name> element contains two other elements (first and last):*

*<name><first>Bill</first><last>Gates</last></name>*

*and the parser will break it up into sub-elements like this:*

*<name>  
  <first>Bill</first>  
  <last>Gates</last>  
</name>*

*Parsed Character Data (PCDATA) is a term used about text data that will be parsed by the XML parser.*

*CDATA - (Unparsed) Character Data*

*The term CDATA is used about text data that should not be parsed by the XML parser.*

*Characters like "<" and "&" are illegal in XML elements.*

*"<" will generate an error because the parser interprets it as the start of a new element.*

*"&" will generate an error because the parser interprets it as the start of an character entity.*

*Some text, like JavaScript code, contains a lot of "<" or "&" characters. To avoid errors script code can be defined as CDATA.*

*// кантэкст выкарыстання CDATA //*

*Everything inside a CDATA section is ignored by the parser.*

*A CDATA section starts with "****<![CDATA[****" and ends with "****]]>****":*

*<script>  
<![CDATA[  
function matchwo(a,b) {  
    if (a < b && a < 0) {  
        return 1;  
    } else {  
        return 0;  
    }  
}  
]]>  
</script>*

*In the example above, everything inside the CDATA section is ignored by the parser.*

***Notes on CDATA sections:***

*A CDATA section cannot contain the string "]]>". Nested CDATA sections are not allowed.*

*The "]]>" that marks the end of the CDATA section cannot contain spaces or line breaks.*

*// А як выкарыстоўваць, напрыклад, скрыпт, змешчаны ў такую секцыю? //*

[*https://www.w3schools.com/xml/dom\_nodes\_navigate.asp*](https://www.w3schools.com/xml/dom_nodes_navigate.asp)

*XML DOM - Navigating Nodes*

*Nodes can be navigated using node relationships.*

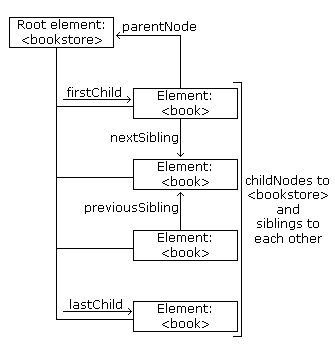
*Navigating DOM Nodes*

*Accessing nodes in the node tree via the relationship between nodes, is often called "navigating nodes".*

*In the XML DOM, node relationships are defined as properties to the nodes:*

* *parentNode*
* *childNodes*
* *firstChild*
* *lastChild*
* *nextSibling*
* *previousSibling*

*The following image illustrates a part of the node tree and the relationship between nodes in*[*books.xml*](javascript:void(0))*:*

**

*DOM - Parent Node*

*All nodes have exactly one parent node. The following code navigates to the parent node of <book>:*

*…*

*Avoid Empty Text Nodes*

*Firefox, and some other browsers, will treat empty white-spaces or new lines as text nodes, Internet Explorer will not.*

*This causes a problem when using the properties: firstChild, lastChild, nextSibling, previousSibling.*

*// апісанне праблемы апрацоўкі white spaces //*

*To avoid navigating to empty text nodes (spaces and new-line characters between element nodes), we use a function that checks the node type:*

*function get\_nextSibling(n) {  
    var y = n.nextSibling;  
    while (y.nodeType! = 1) {  
        y = y.nextSibling;  
    }  
    return y;  
}*

*The function above allows you to use get\_nextSibling(node) instead of the property node.nextSibling.*

*Code explained:*

*Element nodes are type 1. If the sibling node is not an element node, it moves to the next nodes until an element node is found. This way, the result will be the same in both Internet Explorer and Firefox.*

*Get the First Child Element*

*…*

*More Examples*

[*lastChild()*](https://www.w3schools.com/xml/tryit.asp?filename=try_dom_nav_lastchild) *This example uses the lastChild() method and a custom function to get the last child node of a node*

[*nextSibling()*](https://www.w3schools.com/xml/tryit.asp?filename=try_dom_nav_nextsibling) *This example uses the nextSibling() method and a custom function to get the next sibling node of a node*

[*previousSibling()*](https://www.w3schools.com/xml/tryit.asp?filename=try_dom_nav_prevsibling) *This example uses the previousSibling() method and a custom function to get the previous sibling node of a node*

[*https://www.w3schools.com/xml/dom\_nodes\_get.asp*](https://www.w3schools.com/xml/dom_nodes_get.asp)

*XML DOM Get Node Values*

*The nodeValue property is used to get the text value of a node.*

*The getAttribute() method returns the value of an attribute.*

*Get the Value of an Element*

*In the DOM, everything is a node. Element nodes do not have a text value.*

*The text value of an element node is stored in a child node. This node is called a text node.*

*To retrieve the text value of an element, you must retrieve the value of the elements' text node.*

*The getElementsByTagName Method*

*The****getElementsByTagName()****method returns a****node list of all elements****, with the specified tag name, in the same order as they appear in the source document.*

*…*

*The ChildNodes Property*

*The****childNodes****property returns a****list of an element's child nodes****.*

*The following code retrieves the text node of the first <title> element:*

*x = xmlDoc.getElementsByTagName("title")[0];  
y = x.childNodes[0];*

*The nodeValue Property*

*The****nodeValue****property returns the****text value of a text node****.*

*The following code retrieves the text value of the text node of the first <title> element:*

*Example*

*x = xmlDoc.getElementsByTagName("title")[0];  
y = x.childNodes[0];  
z = y.nodeValue;*

*Resul in z: "Everyday Italian"*

*Complete Example*

*…*

*Get the Value of an Attribute*

*In the DOM, attributes are nodes. Unlike element nodes, attribute nodes have text values.*

*The way to get the value of an attribute, is to get its text value.*

*This can be done using the getAttribute() method or using the nodeValue property of the attribute node.*

*Get an Attribute Value - getAttribute()*

*The****getAttribute()****method returns an****attribute's value****.*

*The following code retrieves the text value of the "lang" attribute of the first <title> element:*

*Example*

*x = xmlDoc.getElementsByTagName("title")[0];  
txt = x.getAttribute("lang");*

*Result in txt: "en"*

*…*

*Get an Attribute Value - getAttributeNode()*

*The****getAttributeNode()****method returns an****attribute node****.*

*The following code retrieves the text value of the "lang" attribute of the first <title> element:*

*…*

[*https://www.w3schools.com/xml/dom\_nodes\_set.asp*](https://www.w3schools.com/xml/dom_nodes_set.asp)

*XML DOM Change Node Values*

*The nodeValue property is used to change a node value.*

*The setAttribute() method is used to change an attribute value.*

*…*

*Change the Value of an Element*

*In the DOM, everything is a node. Element nodes do not have a text value.*

*The text value of an element node is stored in a child node. This node is called a text node.*

*To change the text value of an element, you must change the value of the elements's text node.*

*Change the Value of a Text Node*

*The****nodeValue****property can be used to change****the value of a text node****.*

*Suppose "*[*books.xml*](javascript:void(0))*" has been loaded into xmlDoc. // важнае дзеянне, задача ці падзея ў тэхналогіі xml //*

*This code changes the text node value of the first <title> element:*

*…*

*Change the Value of an Attribute*

*In the DOM, attributes are nodes. Unlike element nodes, attribute nodes have text values.*

*The way to change the value of an attribute, is to change its text value.*

*This can be done using the setAttribute() method or setting the nodeValue property of the attribute node.*

*Change an Attribute Using setAttribute()*

*The****setAttribute()****method****changes the value of an attribute****.*

*If the attribute does not exist, a new attribute is created.*

*This code changes the category attribute of the <book> element:*

*…*

***Note:****If the attribute does not exist, a new attribute is created (with the name and value specified).*

*Change an Attribute Using nodeValue*

*The****nodeValue****property is****the value of a attribute node****.*

*Changing the value property changes the value of the attribute.*

*…*

[*https://www.w3schools.com/xml/dom\_nodes\_remove.asp*](https://www.w3schools.com/xml/dom_nodes_remove.asp)

*XML DOM Remove Nodes*

*The removeChild() method removes a specified node.*

*The removeAttribute() method removes a specified attribute.*

*Remove an Element Node*

*The****removeChild()****method removes a specified node.*

*When a node is removed, all its child nodes are also removed.*

*This code will remove the first <book> element from the loaded xml:*

*Example*

*y = xmlDoc.getElementsByTagName("book")[0];  
  
xmlDoc.documentElement.removeChild(y);*

*Example explained:*

1. *Suppose "*[*books.xml*](javascript:void(0))*" is loaded xmlDoc*
2. *Set the variable y to be the element node to remove*
3. *Remove the element node by using the removeChild() method from the parent node*

*Remove Myself - Remove the Current Node*

*The removeChild() method is the only way to remove a specified node.*

*When you have navigated to the node you want to remove, it is possible to remove that node using the parentNode property and the removeChild() method:*

*Example*

*x = xmlDoc.getElementsByTagName("book")[0];  
  
x.parentNode.removeChild(x);*

*Example explained:*

1. *Suppose "*[*books.xml*](javascript:void(0))*" is loaded into xmlDoc*
2. *Set the variable y to be the element node to remove*
3. *Remove the element node by using the parentNode property and the removeChild() method*

*Remove a Text Node*

*The****removeChild()****method can also be used to remove a text node:*

*Example*

*x = xmlDoc.getElementsByTagName("title")[0];  
y = x.childNodes[0];  
x.removeChild(y);*

*Example explained:*

1. *Suppose "*[*books.xml*](javascript:void(0))*" is loaded into xmlDoc*
2. *Set the variable x to be the first title element node*
3. *Set the variable y to be the text node to remove*
4. *Remove the element node by using the removeChild() method from the parent node*

*It is not very common to use removeChild() just to remove the text from a node. The nodeValue property can be used instead. See next paragraph.*

*Clear a Text Node (nodeValue)*

*The****nodeValue****property can be used to change the value of a text node:*

*Example*

*xmlDoc.getElementsByTagName("title")[0].childNodes[0].nodeValue = "";*

[*…*](https://www.w3schools.com/xml/tryit.asp?filename=try_dom_remove_nodevalue)

*Remove an Attribute Node by Name*

*The****removeAttribute(****) method removes an attribute node by its name.*

*Example: removeAttribute('category')*

*This code removes the "category" attribute in the first <book> element:*

*Example*

*x = xmlDoc.getElementsByTagName("book");  
x[0].removeAttribute("category");*

*Example explained:*

1. *Suppose "*[*books.xml*](javascript:void(0))*" is loaded into xmlDoc*
2. *Use getElementsByTagName() to get book nodes*
3. *Remove the "category" attribute form the first book element node*

*Remove Attribute Nodes by Object*

*The****removeAttributeNode(****) method removes an attribute node, using the node object as parameter.*

*Example: removeAttributeNode(x)*

*This code removes all the attributes of all <book> elements:*

*Example*

*x = xmlDoc.getElementsByTagName("book");  
  
for (i = 0; i < x.length; i++) {  
    while (x[i].attributes.length > 0) {  
        attnode = x[i].attributes[0];  
        old\_att = x[i].removeAttributeNode(attnode);  
    }  
}*

*Example explained:*

1. *Suppose "*[*books.xml*](javascript:void(0))*" is loaded into xmlDoc*
2. *Use getElementsByTagName() to get all book nodes*
3. *For each book element check if there are any attributes*
4. *While there are attributes in a book element, remove the attribute*

[*https://www.w3schools.com/xml/dom\_nodes\_replace.asp*](https://www.w3schools.com/xml/dom_nodes_replace.asp)

*XML DOM Replace Nodes*

*The replaceChild() method replaces a specified node.*

*The nodeValue property replaces text in a text node.*

*Replace an Element Node*

*The replaceChild() method is used to replace a node.*

*The following code fragment replaces the first <book> element:*

*…*

*Replace Data In a Text Node*

*The replaceData() method is used to replace data in a text node.*

*The replaceData() method has three parameters:*

* *offset - Where to begin replacing characters. Offset value starts at zero*
* *length - How many characters to replace*
* *string - The string to insert*

*Example*

*xmlDoc=loadXMLDoc("books.xml");  
  
x=xmlDoc.getElementsByTagName("title")[0].childNodes[0];  
  
x.replaceData(0,8,"Easy");*

*Example explained:*

1. *Load "*[*books.xml*](javascript:void(0))*" into xmlDoc*
2. *Get the text node of the first <title> element node*
3. *Use the replaceData method to replace the eight first characters from the text node with "Easy"*

*Use the nodeValue Property Instead*

*It is easier to replace the data in a text node using the nodeValue property.*

*The following code fragment will replace the text node value in the first <title> element with "Easy Italian":*

*…*

*You can read more about changing node values in the*[*Change Node chapter*](https://www.w3schools.com/xml/dom_nodes_set.asp)*. // графавая структура рэпр //*

[*https://www.w3schools.com/xml/dom\_nodes\_create.asp*](https://www.w3schools.com/xml/dom_nodes_create.asp)

*XML DOM Create Nodes*

*Create a New Element Node*

*The****createElement()****method creates a new element node:*

*Example*

*newElement = xmlDoc.createElement("edition");  
  
xmlDoc.getElementsByTagName("book")[0].appendChild(newElement);*

*Example explained:*

1. *Suppose "*[*books.xml*](javascript:void(0))*" is loaded into xmlDoc*
2. *Create a new element node <edition>*
3. *Append the element node to the first <book> element*

*…*

*Create a New Attribute Node*

*The****createAttribute()****is used to create a new attribute node:*

*Example*

*newAtt = xmlDoc.createAttribute("edition");  
newAtt.nodeValue = "first";  
  
xmlDoc.getElementsByTagName("title")[0].setAttributeNode(newAtt);*

*Example explained:*

1. *Suppose "*[*books.xml*](javascript:void(0))*" is loaded into xmlDoc*
2. *Create a new attribute node "edition"*
3. *Set the value of the attribute node to "first"*
4. *Add the new attribute node to the first <title> element*

*…*

*If the attribute already exists, it is replaced by the new one.*

*Create an Attribute Using setAttribute()*

*Since the****setAttribute()****method creates a new attribute if the attribute does not exist, it can be used to create a new attribute.*

*Example*

*xmlDoc.getElementsByTagName('book')[0].setAttribute("edition","first");*

*Example explained:*

1. *Suppose "*[*books.xml*](javascript:void(0))*" is loaded into xmlDoc*
2. *Set the attribute "edition" value to "first" for the first <book> element*

*Loop through all <title> elements and add a new attribute:*[*Try it yourself*](https://www.w3schools.com/xml/tryit.asp?filename=try_dom_createattribute4)

*Create a Text Node*

*The****createTextNode()****method creates a new text node:*

*Example*

*newEle = xmlDoc.createElement("edition");  
newText = xmlDoc.createTextNode("first");  
newEle.appendChild(newText);  
  
xmlDoc.getElementsByTagName("book")[0].appendChild(newEle);*

*Example explained:*

1. *Suppose "*[*books.xml*](javascript:void(0))*" is loaded into xmlDoc*
2. *Create a new element node <edition>*
3. *Create a new text node with the text "first"*
4. *Append the new text node to the element node*
5. *Append the new element node to the first <book> element*

*…*

*Create a CDATA Section Node*

*The****createCDATASection()****method creates a new CDATA section node.*

*Example*

*newCDATA = xmlDoc.createCDATASection("Special Offer & Book Sale");  
  
xmlDoc.getElementsByTagName("book")[0].appendChild(newCDATA);*

*Example explained:*

1. *Suppose "*[*books.xml*](javascript:void(0))*" is loaded into xmlDoc*
2. *Create a new CDATA section node*
3. *Append the new CDATA node to the first <book> element*

*Loop through, and add a CDATA section, to all <book> elements:*[*Try it yourself*](https://www.w3schools.com/xml/tryit.asp?filename=try_dom_createcdatasection)

*Create a Comment Node*

*The****createComment()****method creates a new comment node.*

*Example*

*newComment = xmlDoc.createComment("Revised March 2015");  
  
xmlDoc.getElementsByTagName("book")[0].appendChild(newComment);*

*Example explained:*

1. *Suppose "*[*books.xml*](javascript:void(0))*" is loaded into xmlDoc using*
2. *Create a new comment node*
3. *Append the new comment node to the first <book> element*

*Loop through, and add a comment node, to all <book> elements:*[*Try it yourself*](https://www.w3schools.com/xml/tryit.asp?filename=try_dom_createcomment)

*// апошнія два параграфы – гэта яшчэ два тыпы вузлоў: сі-дэйта секцыі і каментары //*

[*https://www.w3schools.com/xml/dom\_nodes\_add.asp*](https://www.w3schools.com/xml/dom_nodes_add.asp)

*XML DOM Add Nodes*

*Add a Node - appendChild()*

*The****appendChild()****method adds a child node to an existing node.*

*The new node is added (appended) after any existing child nodes.*

***Note:****Use insertBefore() if the position of the node is important.*

*…*

*Insert a Node - insertBefore()*

*The****insertBefore()****method inserts a node before a specified child node.*

*This method is useful when the position of the added node is important:*

*Example*

*newNode = xmlDoc.createElement("book");  
  
x = xmlDoc.documentElement;  
y = xmlDoc.getElementsByTagName("book")[3];  
  
x.insertBefore(newNode,y);*

*Example explained:*

1. *Suppose "*[*books.xml*](javascript:void(0))*" is loaded into xmlDoc*
2. *Create a new element node <book>*
3. *Insert the new node in front of the last <book> element node*

*If the second parameter of insertBefore() is null, the new node will be added after the last existing child node.*

***x.insertBefore(newNode,null)****and****x.appendChild(newNode)****will both append a new child node to x.*

*Add a New Attribute*

*The****setAttribute()****method sets the value of an attribute.*

*…*

*There is no method called add Attribute()  
The setAttribute() will create a new attribute if the attribute does not exist.*

***Note:****If the attribute already exists, the setAttribute() method will overwrite the existing value.*

*Add Text to a Text Node - insertData()*

*The****insertData()****method inserts data into an existing text node.*

*The insertData() method has two parameters:*

* *offset - Where to begin inserting characters (starts at zero)*
* *string - The string to insert*

*…*

[*https://www.w3schools.com/xml/dom\_nodes\_clone.asp*](https://www.w3schools.com/xml/dom_nodes_clone.asp)

*XML DOM Clone Nodes*

*Copy a Node*

*The****cloneNode()****method creates a copy of a specified node.*

*The cloneNode() method has a parameter (true or false). This parameter indicates if the cloned node should include all attributes and child nodes of the original node.*

*The following code fragment copies the first <book> node and appends it to the root node of the document:*

*Example*

*oldNode = xmlDoc.getElementsByTagName('book')[0];  
newNode = oldNode.cloneNode(true);  
xmlDoc.documentElement.appendChild(newNode);*

*Result:*

*Everyday Italian  
Harry Potter  
XQuery Kick Start  
Learning XML  
Everyday Italian*

*Example explained:*

1. *Suppose "*[*books.xml*](javascript:void(0))*" is loaded into xmlDoc*
2. *Get the node to copy (oldNode)*
3. *Clone the node into "newNode"*
4. *Append the new node to the the root node of the XML document*

[*https://www.w3schools.com/xml/dom\_examples.asp*](https://www.w3schools.com/xml/dom_examples.asp)

*тут пералічаны ўсе прыклады*

*Дакументацыя па DOM*

[*https://www.w3schools.com/xml/dom\_nodetype.asp*](https://www.w3schools.com/xml/dom_nodetype.asp)

[*https://www.w3schools.com/xml/dom\_node.asp*](https://www.w3schools.com/xml/dom_node.asp)

[*https://www.w3schools.com/xml/dom\_nodelist.asp*](https://www.w3schools.com/xml/dom_nodelist.asp)

[*https://www.w3schools.com/xml/dom\_namednodemap.asp*](https://www.w3schools.com/xml/dom_namednodemap.asp)

[*https://www.w3schools.com/xml/dom\_document.asp*](https://www.w3schools.com/xml/dom_document.asp)

[*https://www.w3schools.com/xml/dom\_element.asp*](https://www.w3schools.com/xml/dom_element.asp)

[*https://www.w3schools.com/xml/dom\_attribute.asp*](https://www.w3schools.com/xml/dom_attribute.asp)

[*https://www.w3schools.com/xml/dom\_text.asp*](https://www.w3schools.com/xml/dom_text.asp)

[*https://www.w3schools.com/xml/dom\_cdatasection.asp*](https://www.w3schools.com/xml/dom_cdatasection.asp)

[*https://www.w3schools.com/xml/dom\_comment.asp*](https://www.w3schools.com/xml/dom_comment.asp)

[*https://www.w3schools.com/xml/dom\_http.asp*](https://www.w3schools.com/xml/dom_http.asp)

[*https://www.w3schools.com/xml/dom\_parser.asp*](https://www.w3schools.com/xml/dom_parser.asp)

*дакументацыя па xslt*

[*https://www.w3schools.com/xml/xsl\_elementref.asp*](https://www.w3schools.com/xml/xsl_elementref.asp)

[*https://www.w3schools.com/xml/xsl\_functions.asp*](https://www.w3schools.com/xml/xsl_functions.asp)

*// пачынаем дэтальна вывучаць XPath //*

*XPath*

[*https://www.w3schools.com/xml/xpath\_intro.asp*](https://www.w3schools.com/xml/xpath_intro.asp)

*XPath Tutorial*

*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* | * *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 Path Expressions*

*XPath uses path expressions to select nodes or node-sets in an XML document.*

*These path expressions look very much like the path expressions you use with traditional computer file systems:*

**

*XPath Standard Functions*

*XPath includes over 200 built-in functions.*

*There are functions for string values, numeric values, booleans, date and time comparison, node manipulation, sequence manipulation, and much more.*

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

*XPath is Used in XSLT*

*XPath is a major element in the XSLT standard.*

*With XPath knowledge you will be able to take great advantage of your XSLT knowledge.*

*XPath is a W3C Recommendation*

*XPath 1.0 became a W3C Recommendation on November 16, 1999.*

*XPath 2.0 became a W3C Recommendation on January 23, 2007.*

*XPath 3.0 became a W3C Recommendation on April 8, 2014.*

*// technology history //*

[*https://www.w3schools.com/xml/xpath\_nodes.asp*](https://www.w3schools.com/xml/xpath_nodes.asp)

*XPath Nodes*

*XPath Terminology*

*Nodes*

*In XPath, there are seven kinds of nodes: element, attribute, text, namespace, processing-instruction, comment, and document nodes.*

*XML documents are treated as trees of nodes. The topmost element of the tree is called the root element.*

*Look at the following XML document:*

*<?xml version="1.0" encoding="UTF-8"?>  
  
<bookstore>  
  <book>  
    <title lang="en">Harry Potter</title>  
    <author>J K. Rowling</author>  
    <year>2005</year>  
    <price>29.99</price>  
  </book>  
</bookstore>*

*Example of nodes in the XML document above:*

*<bookstore> (root element node)  
  
<author>J K. Rowling</author> (element node)  
  
lang="en" (attribute node)*

*Atomic values*

*Atomic values are nodes with no children or parent.*

*Example of atomic values:*

*J K. Rowling  
  
"en"*

*Items*

*Items are atomic values or nodes.*

*Relationship of Nodes*

*Parent*

*Each element and attribute has one parent.*

*In the following example; the book element is the parent of the title, author, year, and price:*

*…*

*Children*

*Element nodes may have zero, one or more children.*

*…*

*Siblings*

*Nodes that have the same parent.*

*…*

*Ancestors*

*A node's parent, parent's parent, etc.*

*Descendants*

*A node's children, children's children, etc.*

*…*

[*https://www.w3schools.com/xml/xpath\_syntax.asp*](https://www.w3schools.com/xml/xpath_syntax.asp)

*XPath Syntax*

*XPath uses path expressions to select nodes or node-sets in an XML document. The node is selected by following a path or steps.*

*The XML Example Document*

*We will use the following XML document in the examples below.*

*<?xml version="1.0" encoding="UTF-8"?>  
  
<bookstore>  
  
<book>  
  <title lang="en">Harry Potter</title>  
  <price>29.99</price>  
</book>  
  
<book>  
  <title lang="en">Learning XML</title>  
  <price>39.95</price>  
</book>  
  
</bookstore>*

*Selecting Nodes*

*XPath uses path expressions to select nodes in an XML document. The node is selected by following a path or steps. The most useful path expressions are listed below:*

|  |  |
| --- | --- |
| ***Expression*** | ***Description*** |
| *nodename* | *Selects all nodes with the name "nodename"* |
| */* | *Selects from the root node* |
| *//* | *Selects nodes in the document from the current node that match the selection no matter where they are* |
| *.* | *Selects the current node* |
| *..* | *Selects the parent of the current node* |
| *@* | *Selects attributes* |

*In the table below we have listed some path expressions and the result of the expressions:*

|  |  |
| --- | --- |
| ***Path Expression*** | ***Result*** |
| *bookstore* | *Selects all nodes with the name "bookstore"* |
| */bookstore* | *Selects the root element bookstore*  ***Note:****If the path starts with a slash ( / ) it always represents an absolute path to an element!* |
| *bookstore/book* | *Selects all book elements that are children of bookstore* |
| *//book* | *Selects all book elements no matter where they are in the document* |
| *bookstore//book* | *Selects all book elements that are descendant of the bookstore element, no matter where they are under the bookstore element* |
| *//@lang* | *Selects all attributes that are named lang* |

*// у чым розніца між трэцім і пятым выпадкамі? //*

*// у тым, што ў трэцім выпадку вяртаюцца толькі дзеці, а ў пятым – усе нашчадкі //*

*Predicates*

*Predicates are used to find a specific node or a node that contains a specific value.*

*Predicates are always embedded in square brackets.*

*In the table below we have listed some path expressions with predicates and the result of the expressions:*

|  |  |
| --- | --- |
| ***Path Expression*** | ***Result*** |
| */bookstore/book[1]* | *Selects the first book element that is the child of the bookstore element.*  ***Note:*** *In IE 5,6,7,8,9 first node is[0], but according to W3C, it is [1]. To solve this problem in IE, set the SelectionLanguage to XPath:*  *In JavaScript: xml.setProperty("SelectionLanguage","XPath");* |
| */bookstore/book[last()]* | *Selects the last book element that is the child of the bookstore element* |
| */bookstore/book[last()-1]* | *Selects the last but one book element that is the child of the bookstore element* |
| */bookstore/book[position()<3]* | *Selects the first two book elements that are children of the bookstore element* |
| *//title[@lang]* | *Selects all the title elements that have an attribute named lang* |
| *//title[@lang='en']* | *Selects all the title elements that have a "lang" attribute with a value of "en"* |
| */bookstore/book[price>35.00]* | *Selects all the book elements of the bookstore element that have a price element with a value greater than 35.00* |
| */bookstore/book[price>35.00]/title* | *Selects all the title elements of the book elements of the bookstore element that have a price element with a value greater than 35.00* |

*// даволі зручная тэхналогія //*

*Selecting Unknown Nodes*

*XPath wildcards can be used to select unknown XML nodes.*

|  |  |
| --- | --- |
| ***Wildcard*** | ***Description*** |
| *\** | *Matches any element node* |
| *@\** | *Matches any attribute node* |
| *node()* | *Matches any node of any kind* |

*In the table below we have listed some path expressions and the result of the expressions:*

|  |  |
| --- | --- |
| ***Path Expression*** | ***Result*** |
| */bookstore/\** | *Selects all the child element nodes of the bookstore element* |
| *//\** | *Selects all elements in the document* |
| *//title[@\*]* | *Selects all title elements which have at least one attribute of any kind* |

*Selecting Several Paths*

*By using the | operator in an XPath expression you can select several paths.*

*In the table below we have listed some path expressions and the result of the expressions:*

|  |  |
| --- | --- |
| ***Path Expression*** | ***Result*** |
| *//book/title | //book/price* | *Selects all the title AND price elements of all book elements* |
| *//title | //price* | *Selects all the title AND price elements in the document* |
| */bookstore/book/title | //price* | *Selects all the title elements of the book element of the bookstore element AND all the price elements in the document* |

[*https://www.w3schools.com/xml/xpath\_axes.asp*](https://www.w3schools.com/xml/xpath_axes.asp)

*XPath Axes*

*The XML Example Document*

*We will use the following XML document in the examples below.*

*<?xml version="1.0" encoding="UTF-8"?>  
  
<bookstore>  
  
<book>  
  <title lang="en">Harry Potter</title>  
  <price>29.99</price>  
</book>  
  
<book>  
  <title lang="en">Learning XML</title>  
  <price>39.95</price>  
</book>  
  
</bookstore>*

*XPath Axes*

*An axis defines a node-set relative to the current node.*

|  |  |
| --- | --- |
| ***AxisName*** | ***Result*** |
| *ancestor* | *Selects all ancestors (parent, grandparent, etc.) of the current node* |
| *ancestor-or-self* | *Selects all ancestors (parent, grandparent, etc.) of the current node and the current node itself* |
| *attribute* | *Selects all attributes of the current node* |
| *child* | *Selects all children of the current node* |
| *descendant* | *Selects all descendants (children, grandchildren, etc.) of the current node* |
| *descendant-or-self* | *Selects all descendants (children, grandchildren, etc.) of the current node and the current node itself* |
| *following* | *Selects everything in the document after the closing tag of the current node* |
| *following-sibling* | *Selects all siblings after the current node* |
| *namespace* | *Selects all namespace nodes of the current node* |
| *parent* | *Selects the parent of the current node* |
| *preceding* | *Selects all nodes that appear before the current node in the document, except ancestors, attribute nodes and namespace nodes* |
| *preceding-sibling* | *Selects all siblings before the current node* |
| *self* | *Selects the current node* |

*Location Path Expression*

*A location path can be absolute or relative.*

*An absolute location path starts with a slash ( / ) and a relative location path does not. In both cases the location path consists of one or more steps, each separated by a slash:*

*An absolute location path:  
  
/step/step/...  
  
A relative location path:  
  
step/step/...*

*Each step is evaluated against the nodes in the current node-set.*

*A step consists of:*

* *an axis (defines the tree-relationship between the selected nodes and the current node)*
* *a node-test (identifies a node within an axis)*
* *zero or more predicates (to further refine the selected node-set)*

*The syntax for a location step is:*

*axisname::nodetest[predicate]*

*Examples*

|  |  |
| --- | --- |
| ***Example*** | ***Result*** |
| *child::book* | *Selects all book nodes that are children of the current node* |
| *attribute::lang* | *Selects the lang attribute of the current node* |
| *child::\** | *Selects all element children of the current node* |
| *attribute::\** | *Selects all attributes of the current node* |
| *child::text()* | *Selects all text node children of the current node* |
| *child::node()* | *Selects all children of the current node* |
| *descendant::book* | *Selects all book descendants of the current node* |
| *ancestor::book* | *Selects all book ancestors of the current node* |
| *ancestor-or-self::book* | *Selects all book ancestors of the current node - and the current as well if it is a book node* |
| *child::\*/child::price* | *Selects all price grandchildren of the current node* |

*// нетрывіяльныя прыклады. Вельмі-вельмі цікава. важна папрактыкавацца з пасам //*

[*https://www.w3schools.com/xml/xpath\_operators.asp*](https://www.w3schools.com/xml/xpath_operators.asp)

*XPath Operators*

*An XPath expression returns either a node-set, a string, a Boolean, or a number.*

*XPath Operators*

*Below is a list of the operators that can be used in XPath expressions:*

|  |  |  |
| --- | --- | --- |
| ***Operator*** | ***Description*** | ***Example*** |
| *|* | *Computes two node-sets* | *//book | //cd* |
| *+* | *Addition* | *6 + 4* |
| *-* | *Subtraction* | *6 - 4* |
| *\** | *Multiplication* | *6 \* 4* |
| *div* | *Division* | *8 div 4* |
| *=* | *Equal* | *price=9.80* |
| *!=* | *Not equal* | *price!=9.80* |
| *<* | *Less than* | *price<9.80* |
| *<=* | *Less than or equal to* | *price<=9.80* |
| *>* | *Greater than* | *price>9.80* |
| *>=* | *Greater than or equal to* | *price>=9.80* |
| *or* | *or* | *price=9.80 or price=9.70* |
| *and* | *and* | *price>9.00 and price<9.90* |
| *mod* | *Modulus (division remainder)* | *5 mod 2* |

*// цікава, арыфметычныя аператары – гэта для таго каб работаць з лікамі ў элементах //*

[*https://www.w3schools.com/xml/xpath\_examples.asp*](https://www.w3schools.com/xml/xpath_examples.asp)

*XPath Examples*

*Let's try to learn some basic XPath syntax by looking at some examples.*

*The XML Example Document*

*We will use the following XML document in the examples below.*

*"books.xml":*

*<?xml version="1.0" encoding="UTF-8"?>  
  
<bookstore>  
  
<book category="cooking">  
  <title lang="en">Everyday Italian</title>  
  <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>  
  
<book category="web">  
  <title lang="en">XQuery Kick Start</title>  
  <author>James McGovern</author>  
  <author>Per Bothner</author>  
  <author>Kurt Cagle</author>  
  <author>James Linn</author>  
  <author>Vaidyanathan Nagarajan</author>  
  <year>2003</year>  
  <price>49.99</price>  
</book>  
  
<book category="web">  
  <title lang="en">Learning XML</title>  
  <author>Erik T. Ray</author>  
  <year>2003</year>  
  <price>39.95</price>  
</book>  
  
</bookstore>*

*Loading the XML Document*

*Using an XMLHttpRequest object to load XML documents is supported in all modern browsers.*

*var xmlhttp = new XMLHttpRequest();*

*Code for older browsers (IE5 and IE6) can be found in the AJAX tutorial.*

*Selecting Nodes*

*Unfortunately, there are different ways of dealing with XPath in different browsers.*

*Chrome, Firefox, Edge, Opera, and Safari use the evaluate() method to select nodes:*

*xmlDoc.evaluate(xpath, xmlDoc, null, XPathResult.ANY\_TYPE,null);*

*Internet Explorer uses the selectNodes() method to select node:*

*xmlDoc.selectNodes(xpath);*

*In our examples we have included code that should work with most major browsers.*

*Select all the titles*

*The following example selects all the title nodes:*

*Example*

*/bookstore/book/title*

*Select the title of the first book*

*The following example selects the title of the first book node under the bookstore element:*

*Example*

*/bookstore/book[1]/title*

*Select all the prices*

*The following example selects the text from all the price nodes:*

*Example*

*/bookstore/book/price[text()]*

*Select price nodes with price>35*

*The following example selects all the price nodes with a price higher than 35:*

*Example*

*/bookstore/book[price>35]/price*

*Select title nodes with price>35*

*The following example selects all the title nodes with a price higher than 35:*

*Example*

*/bookstore/book[price>35]/title*

*Пачынаем вывучаць xslt*

*XSLT*

[*https://www.w3schools.com/xml/xsl\_intro.asp*](https://www.w3schools.com/xml/xsl_intro.asp)

*XSLT Introduction*

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

*XSLT stands for XSL Transformations.*

*This tutorial will teach you how to use XSLT to transform XML documents into other formats (like transforming XML into HTML).*

*Online XSLT Editor*

*With our online editor, you can edit XML and XSLT code, and click on a button to view the result.*

*XSLT Example*

*<?xml version="1.0"?>  
  
<xsl:stylesheet version="1.0"  
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">  
  
<xsl:template match="/">  
  <html>  
  <body>  
    <h2>My CD Collection</h2>  
    <table border="1">  
      <tr bgcolor="#9acd32">  
        <th>Title</th>  
        <th>Artist</th>  
      </tr>  
      <xsl:for-each select="catalog/cd">  
        <tr>  
          <td><xsl:value-of select="title"/></td>  
          <td><xsl:value-of select="artist"/></td>  
        </tr>  
      </xsl:for-each>  
    </table>  
  </body>  
  </html>  
</xsl:template>  
  
</xsl:stylesheet>*

*What You Should Already Know*

*Before you continue you should have a basic understanding of the following:*

* *HTML*
* *XML*

*If you want to study these subjects first, find the tutorials on our*[*Home page*](https://www.w3schools.com/default.asp)*.*

*XSLT References*

[*XSLT Elements*](https://www.w3schools.com/xml/xsl_elementref.asp)

*Description of all the XSLT elements from the W3C Recommendation, and information about browser support.*

[*XSLT, XPath, and XQuery Functions*](https://www.w3schools.com/xml/xsl_functions.asp)

*XSLT 2.0, XPath 2.0, and XQuery 1.0, share the same functions library. There are over 100 built-in functions. There are functions for string values, numeric values, date and time comparison, node and QName manipulation, sequence manipulation, and more.*

[*https://www.w3schools.com/xml/xsl\_languages.asp*](https://www.w3schools.com/xml/xsl_languages.asp)

*XSL(T) Languages*

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

***XPath****is a language for navigating in XML documents.*

***XQuery****is a language for querying XML documents. // лепей зразумець //*

*It Started with XSL*

*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.*

*XSL - More Than a Style Sheet Language*

*XSL consists of four parts:*

* *XSLT - a language for transforming XML documents*
* *XPath - a language for navigating in XML documents*
* *XSL-FO - a language for formatting XML documents (discontinued in 2013)*
* *XQuery - a language for querying XML documents*

*With the****CSS3 Paged Media Module****, W3C has delivered a new standard for document formatting. So, since 2013, CSS3 is proposed as an XSL-FO replacement.*

*What is XSLT?*

* *XSLT stands for XSL Transformations*
* *XSLT is the most important part of XSL*
* *XSLT transforms an XML document into another XML document*
* *XSLT uses XPath to navigate in XML documents*
* *XSLT is a W3C Recommendation*

*XSLT = XSL Transformations*

*XSLT is the most important part of XSL.*

*XSLT is used to transform an XML document into another XML document, or another type of document that is recognized by a browser, like HTML and XHTML. Normally XSLT does this by transforming each XML element into an (X)HTML element.*

*With XSLT you can add/remove elements and attributes to or from the output file. You can also rearrange and sort elements, perform tests and make decisions about which elements to hide and display, and a lot more.*

*A common way to describe the transformation process is to say that****XSLT transforms an XML source-tree into an XML result-tree****.*

*XSLT Uses XPath*

*XSLT uses XPath to find information in an XML document. XPath is used to navigate through elements and attributes in XML documents.*

*If you want to study XPath first, please read our*[*XPath Tutorial*](https://www.w3schools.com/xml/xpath_intro.asp)*.*

*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.*

*XSLT Browser Support*

*All major browsers support XSLT and XPath.*

*XSLT is a W3C Recommendation*

*XSLT became a W3C Recommendation 16. November 1999.*

*// праверыць, ці не пісалі дзесьці, што XPath не падтрымліваецца браўзерамі //*

*// не пісалі. Гэта пісалі пра XPointer і XLink //*

[*https://www.w3schools.com/xml/xsl\_transformation.asp*](https://www.w3schools.com/xml/xsl_transformation.asp)

*XSLT - Transformation*

*Example study: How to transform XML into XHTML using XSLT?*

*The details of this example will be explained in the next chapter.*

*Correct Style Sheet Declaration*

*The root element that declares the document to be an XSL style sheet is <xsl:stylesheet> or <xsl:transform>.*

***Note:****<xsl:stylesheet> and <xsl:transform> are completely synonymous and either can be used!*

*The correct way to declare an XSL style sheet according to the W3C XSLT Recommendation is:*

*<xsl:stylesheet version="1.0"  
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">*

*or:*

*<xsl:transform version="1.0"  
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">*

*To get access to the XSLT elements, attributes and features we must declare the XSLT namespace at the top of the document.*

*The xmlns:xsl="http://www.w3.org/1999/XSL/Transform" points to the official W3C XSLT namespace. If you use this namespace, you must also include the attribute version="1.0".*

*Start with a Raw XML Document*

*We want to****transform****the following XML document ("cdcatalog.xml") into XHTML:*

*<?xml version="1.0" encoding="UTF-8"?>  
<catalog>  
  <cd>  
    <title>Empire Burlesque</title>  
    <artist>Bob Dylan</artist>  
    <country>USA</country>  
    <company>Columbia</company>  
    <price>10.90</price>  
    <year>1985</year>  
  </cd>  
.  
.  
</catalog>*

***Viewing XML Files in IE, Chrome, Firefox, Safari, and Opera:****Open the XML file (click on the link below) - The XML document will be displayed with color-coded root and child elements (except in Safari). Often, there is a plus (+) or minus sign (-) to the left of the elements that can be clicked to expand or collapse the element structure.****Tip: To view the raw XML source, right-click in XML file and select "View Source"!***

*Create an XSL Style Sheet*

*Then you create an XSL Style Sheet ("cdcatalog.xsl") with a transformation template:*

*<?xml version="1.0" encoding="UTF-8"?>  
  
<xsl:stylesheet version="1.0"  
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">  
  
<xsl:template match="/">  
  <html>  
  <body>  
  <h2>My CD Collection</h2>  
  <table border="1">  
    <tr bgcolor="#9acd32">  
      <th>Title</th>  
      <th>Artist</th>  
    </tr>  
    <xsl:for-each select="catalog/cd">  
    <tr>  
      <td><xsl:value-of select="title"/></td>  
      <td><xsl:value-of select="artist"/></td>  
    </tr>  
    </xsl:for-each>  
  </table>  
  </body>  
  </html>  
</xsl:template>  
  
</xsl:stylesheet>*

*Link the XSL Style Sheet to the XML Document*

*Add the XSL style sheet reference to your XML document ("cdcatalog.xml"):*

*<?xml version="1.0" encoding="UTF-8"?>  
<?xml-stylesheet type="text/xsl" href="cdcatalog.xsl"?> // гэта processing instruction //  
<catalog>  
  <cd>  
    <title>Empire Burlesque</title>  
    <artist>Bob Dylan</artist>  
    <country>USA</country>  
    <company>Columbia</company>  
    <price>10.90</price>  
    <year>1985</year>  
  </cd>  
.  
.  
</catalog>*

*If you have an XSLT compliant browser it will nicely****transform****your XML into XHTML.*

*The details of the example above will be explained in the next chapters.*

[*https://www.w3schools.com/xml/xsl\_templates.asp*](https://www.w3schools.com/xml/xsl_templates.asp)

*XSLT <xsl:template> Element*

*An XSL style sheet consists of one or more set of rules that are called templates.*

*A template contains rules to apply when a specified node is matched.*

*The <xsl:template> Element*

*The <xsl:template> element is used to build templates.*

*The****match****attribute is used to associate a template with an XML element. The match attribute can also be used to define a template for the entire XML document. The value of the match attribute is an XPath expression (i.e. match="/" defines the whole document).*

*Ok, let's look at a simplified version of the XSL file from the previous chapter:*

*Example*

*<?xml version="1.0" encoding="UTF-8"?>  
<xsl:stylesheet version="1.0"  
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">  
  
<xsl:template match="/">  
  <html>  
  <body>  
  <h2>My CD Collection</h2>  
  <table border="1">  
    <tr bgcolor="#9acd32">  
      <th>Title</th>  
      <th>Artist</th>  
    </tr>  
    <tr>  
      <td>.</td>  
      <td>.</td>  
    </tr>  
  </table>  
  </body>  
  </html>  
</xsl:template>  
  
</xsl:stylesheet>*

*Example Explained*

*Since an XSL style sheet is an XML document, it always begins with the XML declaration:****<?xml version="1.0" encoding="UTF-8"?>****.*

*The next element,****<xsl:stylesheet>****,**defines that this document is an XSLT style sheet document (along with the version number and XSLT namespace attributes).*

*The****<xsl:template>****element defines a template. The****match="/"****attribute associates the template with the root of the XML source document.*

*The content inside the <xsl:template> element defines some HTML to write to the output.*

*The last two lines define the end of the template and the end of the style sheet.*

*The result from this example was a little disappointing, because no data was copied from the XML document to the output. In the next chapter you will learn how to use the****<xsl:value-of>****element to select values from the XML elements.*

[*https://www.w3schools.com/xml/xsl\_value\_of.asp*](https://www.w3schools.com/xml/xsl_value_of.asp)

*XSLT <xsl:value-of> Element*

*The <xsl:value-of> element is used to extract the value of a selected node.*

*The <xsl:value-of> Element*

*The <xsl:value-of> element can be used to extract the value of an XML element and add it to the output stream of the transformation:*

*Example*

*<?xml version="1.0" encoding="UTF-8"?>  
<xsl:stylesheet version="1.0"  
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">  
  
<xsl:template match="/">  
  <html>  
  <body>  
  <h2>My CD Collection</h2>  
  <table border="1">  
    <tr bgcolor="#9acd32">  
      <th>Title</th>  
      <th>Artist</th>  
    </tr>  
    <tr>  
      <td><xsl:value-of select="catalog/cd/title"/></td>  
      <td><xsl:value-of select="catalog/cd/artist"/></td>  
    </tr>  
  </table>  
  </body>  
  </html>  
</xsl:template>  
  
</xsl:stylesheet>*

*Example Explained*

***Note:****The****select****attribute, in the example above, contains an XPath expression. An XPath expression works like navigating a file system; a forward slash (/) selects subdirectories.*

*The result from the example above was a little disappointing; only one line of data was copied from the XML document to the output. In the next chapter you will learn how to use the****<xsl:for-each>****element to loop through the XML elements, and display all of the records.*

*// цікавы прыклад трэпр. Кожны наступны прыклад цудоўна сшыты з мінулым //*

[*https://www.w3schools.com/xml/xsl\_for\_each.asp*](https://www.w3schools.com/xml/xsl_for_each.asp)

*XSLT <xsl:for-each> Element*

*The <xsl:for-each> element allows you to do looping in XSLT.*

*The <xsl:for-each> Element*

*The XSL <xsl:for-each> element can be used to select every XML element of a specified node-set:*

*Example*

*<?xml version="1.0" encoding="UTF-8"?>  
<xsl:stylesheet version="1.0"  
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">  
  
<xsl:template match="/">  
  <html>  
  <body>  
  <h2>My CD Collection</h2>  
  <table border="1">  
    <tr bgcolor="#9acd32">  
      <th>Title</th>  
      <th>Artist</th>  
    </tr>  
    <xsl:for-each select="catalog/cd">  
    <tr>  
      <td><xsl:value-of select="title"/></td>  
      <td><xsl:value-of select="artist"/></td>  
    </tr>  
    </xsl:for-each>  
  </table>  
  </body>  
  </html>  
</xsl:template>  
  
</xsl:stylesheet>*

***Note:****The value of the****select****attribute is an XPath expression. An XPath expression works like navigating a file system; where a forward slash (/) selects subdirectories.*

*Filtering the Output*

*We can also filter the output from the XML file by adding a criterion to the select attribute in the <xsl:for-each> element.*

***<xsl:for-each select="catalog/cd[artist='Bob Dylan']">***

*Legal filter operators are:*

* *=  (equal)*
* *!= (not equal)*
* *&lt; less than*
* *&gt; greater than*

*Take a look at the adjusted XSL style sheet:*

*Example*

*<?xml version="1.0" encoding="UTF-8"?>  
<xsl:stylesheet version="1.0"  
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">  
  
<xsl:template match="/">  
  <html>  
  <body>  
  <h2>My CD Collection</h2>  
  <table border="1">  
    <tr bgcolor="#9acd32">  
      <th>Title</th>  
      <th>Artist</th>  
    </tr>  
    <xsl:for-each select="catalog/cd[artist='Bob Dylan']">  
    <tr>  
      <td><xsl:value-of select="title"/></td>  
      <td><xsl:value-of select="artist"/></td>  
    </tr>  
    </xsl:for-each>  
  </table>  
  </body>  
  </html>  
</xsl:template>  
  
</xsl:stylesheet>*

[*https://www.w3schools.com/xml/xsl\_sort.asp*](https://www.w3schools.com/xml/xsl_sort.asp)

*XSLT <xsl:sort> Element*

*The <xsl:sort> element is used to sort the output.*

*Where to put the Sort Information*

*To sort the output, simply add an <xsl:sort> element inside the <xsl:for-each> element in the XSL file:*

*Example*

*<?xml version="1.0" encoding="UTF-8"?>  
<xsl:stylesheet version="1.0"  
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">  
  
<xsl:template match="/">  
  <html>  
  <body>  
  <h2>My CD Collection</h2>  
  <table border="1">  
    <tr bgcolor="#9acd32">  
      <th>Title</th>  
      <th>Artist</th>  
    </tr>  
    <xsl:for-each select="catalog/cd">  
      <xsl:sort select="artist"/>  
      <tr>  
        <td><xsl:value-of select="title"/></td>  
        <td><xsl:value-of select="artist"/></td>  
      </tr>  
    </xsl:for-each>  
  </table>  
  </body>  
  </html>  
</xsl:template>  
  
</xsl:stylesheet>*

***Note:****The****select****attribute indicates what XML element to sort on.*

[*https://www.w3schools.com/xml/xsl\_if.asp*](https://www.w3schools.com/xml/xsl_if.asp)

*XSLT <xsl:if> Element*

*The <xsl:if> element is used to put a conditional test against the content of the XML file.*

*The <xsl:if> Element*

*To put a conditional if test against the content of the XML file, add an <xsl:if> element to the XSL document.*

*Syntax*

*<xsl:if test="expression">  
  ...some output if the expression is true...  
</xsl:if>*

*Where to Put the <xsl:if> Element*

*To add a conditional test, add the <xsl:if> element inside the <xsl:for-each> element in the XSL file:*

*Example*

*<?xml version="1.0" encoding="UTF-8"?>  
<xsl:stylesheet version="1.0"  
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">  
  
<xsl:template match="/">  
  <html>  
  <body>  
  <h2>My CD Collection</h2>  
  <table border="1">  
    <tr bgcolor="#9acd32">  
      <th>Title</th>  
      <th>Artist</th>  
      <th>Price</th>  
    </tr>  
    <xsl:for-each select="catalog/cd">  
      <xsl:if test="price &gt; 10">  
        <tr>  
          <td><xsl:value-of select="title"/></td>  
          <td><xsl:value-of select="artist"/></td>  
          <td><xsl:value-of select="price"/></td>  
        </tr>  
      </xsl:if>  
    </xsl:for-each>  
  </table>  
  </body>  
  </html>  
</xsl:template>  
  
</xsl:stylesheet>*

***Note:****The value of the required****test****attribute contains the expression to be evaluated.*

*The code above will only output the title and artist elements of the CDs that has a price that is higher than 10.*

[*https://www.w3schools.com/xml/xsl\_choose.asp*](https://www.w3schools.com/xml/xsl_choose.asp)

*XSLT <xsl:choose> Element*

*The <xsl:choose> element is used in conjunction with <xsl:when> and <xsl:otherwise> to express multiple conditional tests.*

*Syntax*

*<xsl:choose>  
  <xsl:when test="expression">  
    ... some output ...  
  </xsl:when>  
  <xsl:otherwise>  
    ... some output ....  
  </xsl:otherwise>  
</xsl:choose>*

*Where to put the Choose Condition*

*To insert a multiple conditional test against the XML file, add the <xsl:choose>, <xsl:when>, and <xsl:otherwise> elements to the XSL file:*

*Example*

*<?xml version="1.0" encoding="UTF-8"?>  
<xsl:stylesheet version="1.0"  
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">  
  
<xsl:template match="/">  
  <html>  
  <body>  
  <h2>My CD Collection</h2>  
  <table border="1">  
    <tr bgcolor="#9acd32">  
      <th>Title</th>  
      <th>Artist</th>  
    </tr>  
    <xsl:for-each select="catalog/cd">  
    <tr>  
      <td><xsl:value-of select="title"/></td>  
      <xsl:choose>  
        <xsl:when test="price &gt; 10">  
          <td bgcolor="#ff00ff">  
          <xsl:value-of select="artist"/></td>  
        </xsl:when>  
        <xsl:otherwise>  
          <td><xsl:value-of select="artist"/></td>  
        </xsl:otherwise>  
      </xsl:choose>  
    </tr>  
    </xsl:for-each>  
  </table>  
  </body>  
  </html>  
</xsl:template>  
  
</xsl:stylesheet>*

*The code above will add a pink background-color to the "Artist" column WHEN the price of the CD is higher than 10.*

*Another Example*

*Here is another example that contains two <xsl:when> elements:*

*Example*

*<?xml version="1.0" encoding="UTF-8"?>  
<xsl:stylesheet version="1.0"  
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">  
  
<xsl:template match="/">  
  <html>  
  <body>  
  <h2>My CD Collection</h2>  
  <table border="1">  
    <tr bgcolor="#9acd32">  
      <th>Title</th>  
      <th>Artist</th>  
    </tr>  
    <xsl:for-each select="catalog/cd">  
    <tr>  
      <td><xsl:value-of select="title"/></td>  
      <xsl:choose>  
        <xsl:when test="price &gt; 10">  
          <td bgcolor="#ff00ff">  
          <xsl:value-of select="artist"/></td>  
        </xsl:when>  
        <xsl:when test="price &gt; 9">  
          <td bgcolor="#cccccc">  
          <xsl:value-of select="artist"/></td>  
        </xsl:when>  
        <xsl:otherwise>  
          <td><xsl:value-of select="artist"/></td>  
        </xsl:otherwise>  
      </xsl:choose>  
    </tr>  
    </xsl:for-each>  
  </table>  
  </body>  
  </html>  
</xsl:template>  
  
</xsl:stylesheet>*

*The code above will add a pink background color to the "Artist" column WHEN the price of the CD is higher than 10, and a grey background-color WHEN the price of the CD is higher than 9 and lower or equal to 10.*

[*https://www.w3schools.com/xml/xsl\_apply\_templates.asp*](https://www.w3schools.com/xml/xsl_apply_templates.asp)

*XSLT <xsl:apply-templates> Element*

*The <xsl:apply-templates> element applies a template to the current element or to the current element's child nodes.*

*If we add a select attribute to the <xsl:apply-templates> element it will process only the child element that matches the value of the attribute. We can use the select attribute to specify the order in which the child nodes are processed.*

*Look at the following XSL style sheet:*

*Example*

*<?xml version="1.0" encoding="UTF-8"?>  
<xsl:stylesheet version="1.0"  
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">  
  
<xsl:template match="/">  
  <html>  
  <body>  
  <h2>My CD Collection</h2>  
  <xsl:apply-templates/>  
  </body>  
  </html>  
</xsl:template>  
  
<xsl:template match="cd">  
  <p>  
  <xsl:apply-templates select="title"/>  
  <xsl:apply-templates select="artist"/>  
  </p>  
</xsl:template>  
  
<xsl:template match="title">  
  Title: <span style="color:#ff0000">  
  <xsl:value-of select="."/></span>  
  <br />  
</xsl:template>  
  
<xsl:template match="artist">  
  Artist: <span style="color:#00ff00">  
  <xsl:value-of select="."/></span>  
  <br />  
</xsl:template>  
  
</xsl:stylesheet>*

*// intbu this example. It is not so understood. There is no loop. How do it work? Я павазіўся і вызначыў, што ён выкарыстоўвае шаблоны пачаргова для элементаў выбаркі, робячы цыкл як бы аўтаматычна //*

*// Сей элемент дазваляе скарыстаць усе тэмплэйты .xsl файла для абранай выбаркі xml-дакумента? Паразважаць //*

[*https://www.w3schools.com/xml/xsl\_client.asp*](https://www.w3schools.com/xml/xsl_client.asp)

*XSLT - On the Client*

*XSLT can be used to transform the document to XHTML in your browser.*

*A JavaScript Solution*

*In the previous chapters we have explained how XSLT can be used to transform a document from XML to XHTML. We did this by adding an XSL style sheet to the XML file and let the browser do the transformation.*

*Even if this works fine, it is not always desirable to include a style sheet reference in an XML file (e.g. it will not work in a non XSLT aware browser.)*

*A more versatile solution would be to use a JavaScript to do the transformation.*

*By using a JavaScript, we can:*

* *do browser-specific testing*
* *use different style sheets according to browser and user needs*

*That is the beauty of XSLT! One of the design goals for XSLT was to make it possible to transform data from one format to another, supporting different browsers and different user needs.*

*The XML File and the XSL File*

*Look at the XML document that you have seen in the previous chapters:*

*<?xml version="1.0" encoding="UTF-8"?>  
<catalog>  
  <cd>  
    <title>Empire Burlesque</title>  
    <artist>Bob Dylan</artist>  
    <country>USA</country>  
    <company>Columbia</company>  
    <price>10.90</price>  
    <year>1985</year>  
  </cd>  
.  
.  
</catalog>*

*And the accompanying XSL style sheet:*

*<?xml version="1.0" encoding="UTF-8"?>  
  
<xsl:stylesheet version="1.0"  
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">  
  
<xsl:template match="/">  
  <h2>My CD Collection</h2>  
  <table border="1">  
    <tr bgcolor="#9acd32">  
      <th style="text-align:left">Title</th>  
      <th style="text-align:left">Artist</th>  
    </tr>  
    <xsl:for-each select="catalog/cd">  
    <tr>  
      <td><xsl:value-of select="title" /></td>  
      <td><xsl:value-of select="artist" /></td>  
    </tr>  
    </xsl:for-each>  
  </table>  
</xsl:template>  
  
</xsl:stylesheet>*

***Notice that the XML file does not have a reference to the XSL file.***

***IMPORTANT:****The above sentence indicates that an XML file could be transformed using many different XSL style sheets. // важны аспект тэхналогіі – ізаляцыя стыля ад зместу xml-файла //*

*Transforming XML to XHTML in the Browser*

*Here is the source code needed to transform the XML file to XHTML on the client:*

*Example*

*<!DOCTYPE html>  
<html>  
<head>  
<script>  
function loadXMLDoc(filename)  
{  
if (window.ActiveXObject)  
  {  
  xhttp = new ActiveXObject("Msxml2.XMLHTTP");  
  }  
else   
  {  
  xhttp = new XMLHttpRequest();  
  }  
xhttp.open("GET", filename, false);  
try {xhttp.responseType = "msxml-document"} catch(err) {} // Helping IE11  
xhttp.send("");  
return xhttp.responseXML;  
}  
  
function displayResult()  
{  
xml = loadXMLDoc("cdcatalog.xml");  
xsl = loadXMLDoc("cdcatalog.xsl");  
// code for IE  
if (window.ActiveXObject || xhttp.responseType == "msxml-document")  
  {  
  ex = xml.transformNode(xsl);  
  document.getElementById("example").innerHTML = ex;  
  }  
// code for Chrome, Firefox, Opera, etc.  
else if (document.implementation && document.implementation.createDocument)  
  {  
  xsltProcessor = new XSLTProcessor();  
  xsltProcessor.importStylesheet(xsl);  
  resultDocument = xsltProcessor.transformToFragment(xml, document);  
  document.getElementById("example").appendChild(resultDocument);  
  }  
}  
</script>  
</head>  
<body onload="displayResult()">  
<div id="example" />  
</body>  
</html>*

***Tip:****If you don't know how to write JavaScript, please study our*[*JavaScript tutorial*](https://www.w3schools.com/js/default.asp)*.*

*Example Explained:*

***The loadXMLDoc() function does the following:***

* *Create an XMLHttpRequest object*
* *Use the open() and send() methods of the XMLHttpRequest object to send a request to a server*
* *Get the response data as XML data*

***The displayResult() function is used to display the XML file styled by the XSL file:***

* *Load XML and XSL files*
* *Test what kind of browser the user has*
* *If Internet Explorer:*
  + *Use the transformNode() method to apply the XSL style sheet to the xml document*
  + *Set the body of the current document (id="example") to contain the styled xml document*
* *If other browsers:*
  + *Create a new XSLTProcessor object and import the XSL file to it*
  + *Use the transformToFragment() method to apply the XSL style sheet to the xml document*
  + *Set the body of the current document (id="example") to contain the styled xml document*

*// яркі прыклад, які паказвае, як адрозніваецца код для розных браўзераў. Гэта значна ўскладняе і ўдаражае веб-распрацоўку //*

[*https://www.w3schools.com/xml/xsl\_server.asp*](https://www.w3schools.com/xml/xsl_server.asp)

*XSLT - On the Server*

*To make XML data available to all kind of browsers, we can transform the XML document on the SERVER and send it  back to the browser as XHTML.*

*A Cross Browser Solution*

*In the previous chapter we explained how XSLT can be used to transform a document from XML to XHTML in the browser. // на кампутары кліента // We used a JavaScript and an XML parser for the transformation. However, this will not work in a browser that doesn't have an XML parser. // структура браўзера //*

*To make XML data available to all kind of browsers, we can transform the XML document on the server and send back to the browser as XHTML.*

*That's another beauty of XSLT. One of the design goals for XSLT was to make it possible to transform data from one format to another on a server, returning readable data to all kinds of browsers. // важнае пытанне веб-распрацоўкі: дзе рабіць такое пераўтварэнне? На серверы ці ў кліенце? Здаецца, сервер – самы лепшы аптымальны варыянт? Ці не? //*

*The XML File and the XSLT File*

*Look at the XML document that you have seen in the previous chapters:*

*<?xml version="1.0" encoding="UTF-8"?>  
<catalog>  
  <cd>  
    <title>Empire Burlesque</title>  
    <artist>Bob Dylan</artist>  
    <country>USA</country>  
    <company>Columbia</company>  
    <price>10.90</price>  
    <year>1985</year>  
  </cd>  
.  
.  
</catalog>*

*And the accompanying XSL style sheet:*

*<?xml version="1.0" encoding="UTF-8"?>  
<xsl:stylesheet version="1.0"  
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">  
  
<xsl:template match="/">  
  <h2>My CD Collection</h2>  
  <table border="1">  
    <tr bgcolor="#9acd32">  
      <th style="text-align:left">Title</th>  
      <th style="text-align:left">Artist</th>  
    </tr>  
    <xsl:for-each select="catalog/cd">  
    <tr>  
      <td><xsl:value-of select="title" /></td>  
      <td><xsl:value-of select="artist" /></td>  
    </tr>  
    </xsl:for-each>  
  </table>  
</xsl:template>  
  
</xsl:stylesheet>*

***Notice that the XML file does not have a reference to the XSL file.***

***IMPORTANT:****The above sentence indicates that an XML file could be transformed using many different XSL style sheets.*

*PHP Code: Transform XML to XHTML on the Server*

*Here is the PHP source code needed to transform the XML file to XHTML on the server:*

*<?php  
// Load XML file  
$xml = new DOMDocument;  
$xml->load('cdcatalog.xml');  
  
// Load XSL file  
$xsl = new DOMDocument;  
$xsl->load('cdcatalog.xsl');  
  
// Configure the transformer  
$proc = new XSLTProcessor;  
  
// Attach the xsl rules  
$proc->importStyleSheet($xsl);  
  
echo $proc->transformToXML($xml);  
?>*

***Tip:****If you don't know how to write PHP, please study our*[*PHP tutorial*](https://www.w3schools.com/php/default.asp)*.*

*ASP Code: Transform XML to XHTML on the Server*

*Here is the ASP source code needed to transform the XML file to XHTML on the server:*

*<%  
'Load XML file  
set xml = Server.CreateObject("Microsoft.XMLDOM")  
xml.async = false  
xml.load(Server.MapPath("cdcatalog.xml"))  
  
'Load XSL file  
set xsl = Server.CreateObject("Microsoft.XMLDOM")  
xsl.async = false  
xsl.load(Server.MapPath("cdcatalog.xsl"))  
  
'Transform file  
Response.Write(xml.transformNode(xsl))  
%>*

[*https://www.w3schools.com/xml/xsl\_editxml.asp*](https://www.w3schools.com/xml/xsl_editxml.asp)

*XSLT - Editing XML*

*Data stored in XML files can be edited from an Internet browser.*

*Open, Edit and Save XML*

*Now, we will show how to open, edit, and save an XML file that is stored on the server.*

*We will use XSL to transform the XML document into an HTML form. The values of the XML elements will be written to HTML input fields in an HTML form. The HTML form is editable. After editing the data, the data is going to be submitted back to the server and the XML file will be updated (we will show code for both PHP and ASP). // example and good practice example //*

*The XML File and the XSL File*

*First, take a look at the XML document ("tool.xml"):*

*<?xml version="1.0" encoding="UTF-8"?>  
<tool>  
  <field id="prodName">  
    <value>HAMMER HG2606</value>  
  </field>  
  <field id="prodNo">  
    <value>32456240</value>  
  </field>  
  <field id="price">  
    <value>$30.00</value>  
  </field>  
</tool>*

*Then, take a look at the following style sheet ("tool.xsl"):*

*<?xml version="1.0" encoding="UTF-8"?>  
<xsl:stylesheet version="1.0"  
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">  
  
<xsl:template match="/">  
  <html>  
  <body>  
  <form method="post" action="edittool.asp">  
  <h2>Tool Information (edit):</h2>  
  <table border="0">  
    <xsl:for-each select="tool/field">  
    <tr>  
      <td><xsl:value-of select="@id"/></td>  
      <td>  
      <input type="text">  
      <xsl:attribute name="id">  
        <xsl:value-of select="@id" />  
      </xsl:attribute>  
      <xsl:attribute name="name">  
        <xsl:value-of select="@id" />  
      </xsl:attribute>  
      <xsl:attribute name="value">  
        <xsl:value-of select="value" />  
      </xsl:attribute>  
      </input>  
      </td>  
    </tr>  
    </xsl:for-each>  
  </table>  
  <br />  
  <input type="submit" id="btn\_sub" name="btn\_sub" value="Submit" />  
  <input type="reset" id="btn\_res" name="btn\_res" value="Reset" />  
  </form>  
  </body>  
  </html>  
</xsl:template>  
  
</xsl:stylesheet>*

*The XSL file above loops through the elements in the XML file and creates one input field for each XML "field" element. The value of the XML "field" element's "id" attribute is added to both the "id" and "name" attributes of each HTML input field. The value of each XML "value" element is added to the "value" attribute of each HTML input field. The result is an editable HTML form that contains the values from the XML file. // потым варта будзе лепей разабрацца, як xsl стварае атрыбуты //*

*Then, we have a second style sheet: "tool\_updated.xsl". This is the XSL file that will be used to display the updated XML data. This style sheet will not result in an editable HTML form, but a static HTML table:*

*<?xml version="1.0" encoding="UTF-8"?>  
<xsl:stylesheet version="1.0"  
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">  
  
<xsl:template match="/">  
  <html>  
  <body>  
  <h2>Updated Tool Information:</h2>  
  <table border="1">  
    <xsl:for-each select="tool/field">  
    <tr>  
      <td><xsl:value-of select="@id" /></td>  
      <td><xsl:value-of select="value" /></td>  
    </tr>  
    </xsl:for-each>  
  </table>  
  </body>  
  </html>  
</xsl:template>  
  
</xsl:stylesheet>*

*The PHP File*

*In the "tool.xsl" file above, change the HTML form's action attribute to "edittool.php".*

*The "edittool.php" page contains two functions: The loadFile() function loads and transforms the XML file for display and the updateFile() function applies the changes to the XML file:*

*<?php  
function loadFile($xml, $xsl)  
{  
$xmlDoc = new DOMDocument();  
$xmlDoc->load($xml);  
  
$xslDoc = new DOMDocument();  
$xslDoc->load($xsl);  
  
$proc = new XSLTProcessor();  
$proc->importStyleSheet($xslDoc);  
echo $proc->transformToXML($xmlDoc);  
}  
  
function updateFile($xml)  
{  
$xmlLoad = simplexml\_load\_file($xml);  
$postKeys = array\_keys($\_POST);  
  
foreach($xmlLoad->children() as $x)  
{   
  foreach($\_POST as $key=>$value)  
  {   
    if($key == $x->attributes())  
    {   
      $x->value = $value;  
    }  
  }  
}   
  
$xmlLoad->asXML($xml);  
loadFile($xml,"tool\_updated.xsl");  
}  
  
if($\_POST["btn\_sub"] == "")  
{  
  loadFile("tool.xml", "tool.xsl");  
}  
else  
{  
  updateFile("tool.xml");  
}  
?>*

***Tip:****If you don't know how to write PHP, please study our*[*PHP tutorial*](https://www.w3schools.com/php/default.asp)*. // вось гэты php код я ўжо зразумеў слабенька //*

***Note:****We are doing the transformation and applying the changes to the XML file on the server. This is a cross-browser solution. The client will only get HTML back from the server - which will work in any browser.*

*The ASP File*

*The HTML form in the "tool.xsl" file above has an action attribute with a value of "edittool.asp".*

*The "edittool.asp" page contains two functions: The loadFile() function loads and transforms the XML file for display and the updateFile() function applies the changes to the XML file:*

*<%  
function loadFile(xmlfile,xslfile)  
Dim xmlDoc,xslDoc  
'Load XML and XSL file  
set xmlDoc = Server.CreateObject("Microsoft.XMLDOM")  
xmlDoc.async = false  
xmlDoc.load(xmlfile)  
set xslDoc = Server.CreateObject("Microsoft.XMLDOM")  
xslDoc.async = false  
xslDoc.load(xslfile)  
'Transform file  
Response.Write(xmlDoc.transformNode(xslDoc))  
end function  
  
function updateFile(xmlfile)  
Dim xmlDoc,rootEl,f  
Dim i  
'Load XML file  
set xmlDoc = Server.CreateObject("Microsoft.XMLDOM")  
xmlDoc.async = false  
xmlDoc.load(xmlfile)  
  
'Set the rootEl variable equal to the root element  
Set rootEl = xmlDoc.documentElement  
  
'Loop through the form collection  
for i = 1 To Request.Form.Count  
  'Eliminate button elements in the form  
  if instr(1,Request.Form.Key(i),"btn\_")=0 then  
    'The selectSingleNode method queries the XML file for a single node  
    'that matches a query. This query requests the value element that is  
    'the child of a field element that has an id attribute which matches  
    'the current key value in the Form Collection. When there is a match -  
    'set the text property equal to the value of the current field in the  
    'Form Collection.  
    set f = rootEl.selectSingleNode("field[@id='" & \_  
    Request.Form.Key(i) & "']/value")  
    f.Text = Request.Form(i)  
  end if  
next  
// гэты код злучае дадзены з форм са зместам xml-файлаў. Здаецца, кожная форма захоўвае такіж ідэнтыфікатар, як і элемент xml-дакумента //*

*'Save the modified XML file  
xmlDoc.save xmlfile  
  
'Release all object references  
set xmlDoc=nothing  
set rootEl=nothing  
set f=nothing  
  
'Load the modified XML file with a style sheet that  
'allows the client to see the edited information  
loadFile xmlfile,server.MapPath("tool\_updated.xsl")  
end function  
  
'If form is submitted, update the XML file and display result  
' - if not, transform the XML file for editing  
if Request.Form("btn\_sub")="" then  
  loadFile server.MapPath("tool.xml"),server.MapPath("tool.xsl")  
else  
  updateFile server.MapPath("tool.xml")  
end if  
%>*

*// трэба мне стварыць свой рэдактар кода. Без гэтых каментараў //*

[*https://www.w3schools.com/xml/xsl\_examples.asp*](https://www.w3schools.com/xml/xsl_examples.asp)

*XSLT Examples*

*…*

*// тут пералічаны прыклады //*

*Пачынаем вывучаць загадачны XQuery*

*XQuery*

[*https://www.w3schools.com/xml/xquery\_intro.asp*](https://www.w3schools.com/xml/xquery_intro.asp)

*XQuery Tutorial*

*What is XQuery?*

*XQuery is to XML what SQL is to databases.*

*XQuery is designed to query XML data.*

*XQuery Example*

*for $x in doc("books.xml")/bookstore/book  
where $x/price>30  
order by $x/title  
return $x/title*

*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 is About Querying XML*

*XQuery is a language for finding and extracting elements and attributes from XML documents.*

*Here is an example of what XQuery could solve:*

*"Select all CD records with a price less than $10 from the CD collection stored in cd\_catalog.xml"*

*XQuery and XPath*

*XQuery 1.0 and XPath 2.0 share the same data model and support the same functions and operators. If you have already studied XPath you will have no problems with understanding XQuery.*

*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*

*XQuery is a W3C Recommendation*

*XQuery is compatible with several W3C standards, such as XML, Namespaces, XSLT, XPath, and XML Schema.*

*XQuery 1.0 became a W3C Recommendation in 2007.*

[*https://www.w3schools.com/xml/xquery\_example.asp*](https://www.w3schools.com/xml/xquery_example.asp)

*XQuery Example*

*Let's learn some basic XQuery by looking at an example.*

*The XML Example Document*

*We will use the following XML document in the examples below.*

*"books.xml":*

*<?xml version="1.0" encoding="UTF-8"?>  
  
<bookstore>  
  
<book category="COOKING">  
  <title lang="en">Everyday Italian</title>  
  <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>*

*How to Select Nodes From "books.xml"?*

*Functions*

*XQuery uses functions to extract data from XML documents.*

*The doc() function is used to open the "books.xml" file:*

*doc("books.xml")*

*Path Expressions*

*XQuery uses path expressions to navigate through elements in an XML document.*

*The following path expression is used to select all the title elements in the "books.xml" file:*

*doc("books.xml")/bookstore/book/title*

*(/bookstore selects the bookstore element, /book selects all the book elements under the bookstore element, and /title selects all the title elements under each book element)*

*The XQuery above will extract the following:*

*<title lang="en">Everyday Italian</title>  
<title lang="en">Harry Potter</title>  
<title lang="en">XQuery Kick Start</title>  
<title lang="en">Learning XML</title>*

*Predicates*

*XQuery uses predicates to limit the extracted data from XML documents.*

*The following predicate is used to select all the book elements under the bookstore element that have a price element with a value that is less than 30:*

*doc("books.xml")/bookstore/book[price<30]*

*The XQuery above will extract the following:*

*<book category="CHILDREN">  
  <title lang="en">Harry Potter</title>  
  <author>J K. Rowling</author>  
  <year>2005</year>  
  <price>29.99</price>  
</book>*

[*https://www.w3schools.com/xml/xquery\_flwor.asp*](https://www.w3schools.com/xml/xquery_flwor.asp)

*XQuery FLWOR Expressions*

*FLWOR (pronounced "flower") is an acronym for "For, Let, Where, Order by, Return".*

* ***For****- selects a sequence of nodes*
* ***Let****- binds a sequence to a variable*
* ***Where****- filters the nodes*
* ***Order by****- sorts the nodes*
* ***Return****- what to return (gets evaluated once for every node)*

*The XML Example Document*

*We will use the "books.xml" document in the examples below (same XML file as in the previous chapter).*

*How to Select Nodes From "books.xml" With FLWOR*

*Look at the following path expression:*

*doc("books.xml")/bookstore/book[price>30]/title*

*The expression above will select all the title elements under the book elements that are under the bookstore element that have a price element with a value that is higher than 30.*

*The following FLWOR expression will select exactly the same as the path expression above:*

*for $x in doc("books.xml")/bookstore/book  
where $x/price>30  
return $x/title*

*The result will be:*

*<title lang="en">XQuery Kick Start</title>  
<title lang="en">Learning XML</title>*

*With FLWOR you can sort the result:*

*for $x in doc("books.xml")/bookstore/book  
where $x/price>30  
order by $x/title  
return $x/title*

*The****for****clause selects all book elements under the bookstore element into a variable called $x.*

*The****where****clause selects only book elements with a price element with a value greater than 30.*

*The****order by****clause defines the sort-order. Will be sort by the title element.*

*The****return****clause specifies what should be returned. Here it returns the title elements.*

*The result of the XQuery expression above will be:*

*<title lang="en">Learning XML</title>  
<title lang="en">XQuery Kick Start</title>*

[*https://www.w3schools.com/xml/xquery\_flwor\_html.asp*](https://www.w3schools.com/xml/xquery_flwor_html.asp)

*XQuery FLWOR + HTML*

*The XML Example Document*

*We will use the "books.xml" document in the examples below (same XML file as in the previous chapters).*

*Present the Result In an HTML List*

*Look at the following XQuery FLWOR expression:*

*for $x in doc("books.xml")/bookstore/book/title  
order by $x  
return $x*

*The expression above will select all the title elements under the book elements that are under the bookstore element, and return the title elements in alphabetical order.*

*Now we want to list all the book-titles in our bookstore in an HTML list. We add <ul> and <li> tags to the FLWOR expression:*

*<ul>  
{  
for $x in doc("books.xml")/bookstore/book/title  
order by $x  
return <li>{$x}</li>  
}  
</ul>*

*The result of the above will be:*

*<ul>  
<li><title lang="en">Everyday Italian</title></li>  
<li><title lang="en">Harry Potter</title></li>  
<li><title lang="en">Learning XML</title></li>  
<li><title lang="en">XQuery Kick Start</title></li>  
</ul>*

*// так JQuery можна выкарыстоўваць для пераўтварэння xml у html? Лепей зразумець. Гэта XSLT? //*

*Now we want to eliminate the title element, and show only the data inside the title element:*

*<ul>  
{  
for $x in doc("books.xml")/bookstore/book/title  
order by $x  
return <li>{data($x)}</li>  
}  
</ul>*

*The result will be (an HTML list):*

*<ul>  
<li>Everyday Italian</li>  
<li>Harry Potter</li>  
<li>Learning XML</li>  
<li>XQuery Kick Start</li>  
</ul>*

[*https://www.w3schools.com/xml/xquery\_terms.asp*](https://www.w3schools.com/xml/xquery_terms.asp)

*XQuery Terms*

*In XQuery, there are seven kinds of nodes: element, attribute, text, namespace, processing-instruction, comment, and document (root) nodes.*

*XQuery Terminology*

*Nodes*

*In XQuery, there are seven kinds of nodes: element, attribute, text, namespace, processing-instruction, comment, and document (root) nodes. XML documents are treated as trees of nodes. The root of the tree is called the document node (or root node).*

*Look at the following XML document:*

*<?xml version="1.0" encoding="UTF-8"?>  
  
<bookstore>  
  
<book>  
  <title lang="en">Harry Potter</title>  
  <author>J K. Rowling</author>  
  <year>2005</year>  
  <price>29.99</price>  
</book>  
  
</bookstore>*

*Example of nodes in the XML document above:*

*<bookstore> (document node)  
  
<author>J K. Rowling</author> (element node)  
  
lang="en" (attribute node)*

*Atomic values*

*Atomic values are nodes with no children or parent.*

*Example of atomic values:*

*J K. Rowling  
  
"en"*

*Items*

*Items are atomic values or nodes.*

*Relationship of Nodes*

*Parent*

*Each element and attribute has one parent.*

*In the following example; the book element is the parent of the title, author, year, and price:*

*<book>  
  <title>Harry Potter</title>  
  <author>J K. Rowling</author>  
  <year>2005</year>  
  <price>29.99</price>  
</book>*

*Children*

*Element nodes may have zero, one or more children.*

*In the following example; the title, author, year, and price elements are all children of the book element:*

*<book>  
  <title>Harry Potter</title>  
  <author>J K. Rowling</author>  
  <year>2005</year>  
  <price>29.99</price>  
</book>*

*Siblings*

*Nodes that have the same parent.*

*In the following example; the title, author, year, and price elements are all siblings:*

*<book>  
  <title>Harry Potter</title>  
  <author>J K. Rowling</author>  
  <year>2005</year>  
  <price>29.99</price>  
</book>*

*Ancestors*

*A node's parent, parent's parent, etc.*

*In the following example; the ancestors of the title element are the book element and the bookstore element:*

*<bookstore>  
  
<book>  
  <title>Harry Potter</title>  
  <author>J K. Rowling</author>  
  <year>2005</year>  
  <price>29.99</price>  
</book>  
  
</bookstore>*

*Descendants*

*A node's children, children's children, etc.*

*In the following example; descendants of the bookstore element are the book, title, author, year, and price elements:*

*<bookstore>  
  
<book>  
  <title>Harry Potter</title>  
  <author>J K. Rowling</author>  
  <year>2005</year>  
  <price>29.99</price>  
</book>  
  
</bookstore>*

[*https://www.w3schools.com/xml/xquery\_syntax.asp*](https://www.w3schools.com/xml/xquery_syntax.asp)

*XQuery Syntax*

*XQuery is case-sensitive and XQuery elements, attributes, and variables must be valid XML names.*

*XQuery Basic Syntax Rules*

*Some basic syntax rules:*

* *XQuery is case-sensitive*
* *XQuery elements, attributes, and variables must be valid XML names*
* *An XQuery string value can be in single or double quotes*
* *An XQuery variable is defined with a $ followed by a name, e.g. $bookstore*
* *XQuery comments are delimited by (: and :), e.g. (: XQuery Comment :)*

*The XML Example Document*

*We will use the "books.xml" document in the examples below (same XML file as in the previous chapters).*

*XQuery Conditional Expressions*

*"If-Then-Else" expressions are allowed in XQuery.*

*Look at the following example:*

*for $x in doc("books.xml")/bookstore/book  
return if ($x/@category="CHILDREN")  
then <child>{data($x/title)}</child>  
else <adult>{data($x/title)}</adult>*

***Notes on the "if-then-else" syntax:****parentheses around the if expression are required. else is required, but it can be just else ().*

*The result of the example above will be:*

*<adult>Everyday Italian</adult>  
<child>Harry Potter</child>  
<adult>XQuery Kick Start</adult>  
<adult>Learning XML</adult>*

*XQuery Comparisons*

*In XQuery there are two ways of comparing values.*

*1. General comparisons: =, !=, <, <=, >, >=*

*2. Value comparisons: eq, ne, lt, le, gt, ge*

*The difference between the two comparison methods are shown below.*

*The following expression returns true if any q attributes have a value greater than 10:*

*$bookstore//book/@q > 10*

*The following expression returns true if there is only one q attribute returned by the expression, and its value is greater than 10. If more than one q is returned, an error occurs:*

*$bookstore//book/@q gt 10*

*// explanation of value comparison //*

[*https://www.w3schools.com/xml/xquery\_add.asp*](https://www.w3schools.com/xml/xquery_add.asp)

*XQuery Adding Elements and Attributes*

*The XML Example Document*

*We will use the "books.xml" document in the examples below (same XML file as in the previous chapters).*

*Adding Elements and Attributes to the Result*

*As we have seen in a previous chapter, we may include elements and attributes from the input document ("books.xml) in the result:*

*for $x in doc("books.xml")/bookstore/book/title  
order by $x  
return $x*

*The XQuery expression above will include both the title element and the lang attribute in the result, like this:*

*<title lang="en">Everyday Italian</title>  
<title lang="en">Harry Potter</title>  
<title lang="en">Learning XML</title>  
<title lang="en">XQuery Kick Start</title>*

*The XQuery expression above returns the title elements the exact same way as they are described in the input document.*

*We now want to add our own elements and attributes to the result!*

*Add HTML Elements and Text*

*Now, we want to add some HTML elements to the result. We will put the result in an HTML list - together with some text:*

*<html>  
<body>  
  
<h1>Bookstore</h1>  
  
<ul>  
{  
for $x in doc("books.xml")/bookstore/book  
order by $x/title  
return <li>{data($x/title)}. Category: {data($x/@category)}</li>  
}  
</ul>  
  
</body>  
</html>*

*The XQuery expression above will generate the following result:*

*<html>  
<body>  
  
<h1>Bookstore</h1>  
  
<ul>  
<li>Everyday Italian. Category: COOKING</li>  
<li>Harry Potter. Category: CHILDREN</li>  
<li>Learning XML. Category: WEB</li>  
<li>XQuery Kick Start. Category: WEB</li>  
</ul>  
  
</body>  
</html>*

*// а куды ўстаўляецца гэта XQuery? Адразу ў XML-дакумент? //*

*Add Attributes to HTML Elements*

*Next, we want to use the category attribute as a class attribute in the HTML list:*

*<html>  
<body>  
  
<h1>Bookstore</h1>  
  
<ul>  
{  
for $x in doc("books.xml")/bookstore/book  
order by $x/title  
return <li class="{data($x/@category)}">{data($x/title)}</li>  
}  
</ul>  
  
</body>  
</html>*

*The XQuery expression above will generate the following result:*

*<html>  
<body>  
<h1>Bookstore</h1>  
  
<ul>  
<li class="COOKING">Everyday Italian</li>  
<li class="CHILDREN">Harry Potter</li>  
<li class="WEB">Learning XML</li>  
<li class="WEB">XQuery Kick Start</li>  
</ul>  
  
</body>  
</html>*

*// відаць, XQuery выкарыстоўваць тады, калі неабходна выцягваць дадзены з xml-дакументаў, але пры гэтым XSLT не патрэбны //*

[*https://www.w3schools.com/xml/xquery\_select.asp*](https://www.w3schools.com/xml/xquery_select.asp)

*XQuery Selecting and Filtering*

*The XML Example Document*

*We will use the "books.xml" document in the examples below (same XML file as in the previous chapters).*

*Selecting and Filtering Elements*

*As we have seen in the previous chapters, we are selecting and filtering elements with either a Path expression or with a FLWOR expression.*

*Look at the following FLWOR expression:*

*for $x in doc("books.xml")/bookstore/book  
where $x/price>30  
order by $x/title  
return $x/title*

* *for - (optional) binds a variable to each item returned by the in expression*
* *let - (optional)*
* *where - (optional) specifies a criteria*
* *order by - (optional) specifies the sort-order of the result*
* *return - specifies what to return in the result*

*The for Clause*

*The for clause binds a variable to each item returned by the in expression. The for clause results in iteration. There can be multiple for clauses in the same FLWOR expression.*

*To loop a specific number of times in a for clause, you may use the****to****keyword:*

*for $x in (1 to 5)  
return <test>{$x}</test>*

*Result:*

*<test>1</test>  
<test>2</test>  
<test>3</test>  
<test>4</test>  
<test>5</test>*

*The****at****keyword can be used to count the iteration:*

*for $x at $i in doc("books.xml")/bookstore/book/title  
return <book>{$i}. {data($x)}</book>*

*Result:*

*<book>1. Everyday Italian</book>  
<book>2. Harry Potter</book>  
<book>3. XQuery Kick Start</book>  
<book>4. Learning XML</book>*

*It is also allowed with more than one in expression in the for clause. Use comma to separate each in expression:*

*for $x in (10,20), $y in (100,200)  
return <test>x={$x} and y={$y}</test>*

*Result:*

*<test>x=10 and y=100</test>  
<test>x=10 and y=200</test>  
<test>x=20 and y=100</test>  
<test>x=20 and y=200</test>*

*The let Clause*

*The let clause allows variable assignments and it avoids repeating the same expression many times. The let clause does not result in iteration.*

*let $x := (1 to 5)  
return <test>{$x}</test>*

*Result:*

*<test>1 2 3 4 5</test>*

*The where Clause*

*The where clause is used to specify one or more criteria for the result:*

*where $x/price>30 and $x/price<100*

*The order by Clause*

*The order by clause is used to specify the sort order of the result. Here we want to order the result by category and title:*

*for $x in doc("books.xml")/bookstore/book  
order by $x/@category, $x/title  
return $x/title*

*Result:*

*<title lang="en">Harry Potter</title>  
<title lang="en">Everyday Italian</title>  
<title lang="en">Learning XML</title>  
<title lang="en">XQuery Kick Start</title>*

*The return Clause*

*The return clause specifies what is to be returned.*

*for $x in doc("books.xml")/bookstore/book  
return $x/title*

*Result:*

*<title lang="en">Everyday Italian</title>  
<title lang="en">Harry Potter</title>  
<title lang="en">XQuery Kick Start</title>  
<title lang="en">Learning XML</title>*

[*https://www.w3schools.com/xml/xquery\_functions.asp*](https://www.w3schools.com/xml/xquery_functions.asp)

*XQuery Functions*

*XQuery 1.0, XPath 2.0, and XSLT 2.0 share the same functions library.*

*XQuery is built on XPath expressions. XQuery 1.0 and XPath 2.0 share the same data model and support the same functions and operators.*

[*XPath Operators*](https://www.w3schools.com/xml/xpath_operators.asp)

[*XPath Functions*](https://www.w3schools.com/xml/xsl_functions.asp)

*You can also define your own functions in XQuery.*

*XQuery Data Types*

*XQuery shares the same data types as XML Schema 1.0 (XSD).*

[*XSD String*](https://www.w3schools.com/xml/schema_dtypes_string.asp)

[*XSD Date*](https://www.w3schools.com/xml/schema_dtypes_date.asp)

[*XSD Numeric*](https://www.w3schools.com/xml/schema_dtypes_numeric.asp)

[*XSD Misc*](https://www.w3schools.com/xml/schema_dtypes_misc.asp)

*Examples of Function Calls*

*A call to a function can appear where an expression may appear. Look at the examples below:*

*Example 1: In an element*

*<name>{upper-case($booktitle)}</name>*

*Example 2: In the predicate of a path expression*

*doc("books.xml")/bookstore/book[substring(title,1,5)='Harry']*

*Example 3: In a let clause*

*let $name := (substring($booktitle,1,4))*

*XQuery User-Defined Functions*

*If you cannot find the XQuery function you need, you can write your own.*

*User-defined functions can be defined in the query or in a separate library.*

*Syntax*

*declare function prefix:function\_name($parameter as datatype)  
as returnDatatype  
{  
 ...function code here...  
};*

*Notes on user-defined functions:*

* *Use the declare function keyword*
* *The name of the function must be prefixed*
* *The data type of the parameters are mostly the same as the data types defined in XML Schema*
* *The body of the function must be surrounded by curly braces*

*Example of a User-defined Function Declared in the Query*

*declare function local:minPrice($p as xs:decimal?,$d as xs:decimal?)  
as xs:decimal?  
{  
let $disc := ($p \* $d) div 100  
return ($p - $disc)  
};  
  
Below is an example of how to call the function above:  
  
<minPrice>{local:minPrice($book/price,$book/discount)}</minPrice>*

*// але не разумею, калі выкарыстоўваць XQuery і чаму яе функцыянальнасць не забяспечваецца іншым? //*

*// пра выкарыстанне чытай вышэй. А як выкарыстоўваць функцыі, напісанныя ў іншых файлах? Як экспартаваць файлы? //*

*DTD*

[*https://www.w3schools.com/xml/xml\_dtd\_intro.asp*](https://www.w3schools.com/xml/xml_dtd_intro.asp)

*DTD пачынаем чытаць!*

*DTD Tutorial*

*What is a DTD?*

*A DTD is a Document Type Definition.*

*A DTD defines the structure and the legal elements and attributes of an XML document.*

*Why Use a DTD?*

*With a DTD, independent groups of people can agree on a standard DTD for interchanging data.*

*An application can use a DTD to verify that XML data is valid.*

*An Internal DTD Declaration*

*If the DTD is declared inside the XML file, it must be wrapped inside the <!DOCTYPE> definition:*

*XML document with an internal DTD*

*<?xml version="1.0"?>  
<!DOCTYPE note [  
<!ELEMENT note (to,from,heading,body)>  
<!ELEMENT to (#PCDATA)>  
<!ELEMENT from (#PCDATA)>  
<!ELEMENT heading (#PCDATA)>  
<!ELEMENT body (#PCDATA)>  
]>  
<note>  
<to>Tove</to>  
<from>Jani</from>  
<heading>Reminder</heading>  
<body>Don't forget me this weekend</body>  
</note>*

*// DTD is actually not visible in browser //*

*The DTD above is interpreted like this:*

* ***!DOCTYPE note****defines that the root element of this document is note*
* ***!ELEMENT note****defines that the note element must contain four elements: "to,from,heading,body"*
* ***!ELEMENT to****defines the to element to be of type "#PCDATA"*
* ***!ELEMENT from****defines the from element to be of type "#PCDATA"*
* ***!ELEMENT heading****defines the heading element to be of type "#PCDATA"*
* ***!ELEMENT body****defines the body element to be of type "#PCDATA"*

*An External DTD Declaration*

*If the DTD is declared in an external file, the <!DOCTYPE> definition must contain a reference to the DTD file:*

*XML document with a reference to an external DTD*

*<?xml version="1.0"?>  
<!DOCTYPE note SYSTEM "note.dtd">  
<note>  
  <to>Tove</to>  
  <from>Jani</from>  
  <heading>Reminder</heading>  
  <body>Don't forget me this weekend!</body>  
</note>*

*And here is the file "note.dtd", which contains the DTD:*

*<!ELEMENT note (to,from,heading,body)>  
<!ELEMENT to (#PCDATA)>  
<!ELEMENT from (#PCDATA)>  
<!ELEMENT heading (#PCDATA)>  
<!ELEMENT body (#PCDATA)>*

[*https://www.w3schools.com/xml/xml\_dtd\_building.asp*](https://www.w3schools.com/xml/xml_dtd_building.asp)

*DTD - XML Building Blocks*

*The main building blocks of both XML and HTML documents are elements.*

*Seen from a DTD point of view, all XML documents are made up by the following building blocks:*

* *Elements*
* *Attributes*
* *Entities*
* *PCDATA*
* *CDATA*

*Elements*

*Elements are the****main building blocks****of both XML and HTML documents.*

*Examples of HTML elements are "body" and "table". Examples of XML elements could be "note" and "message". Elements can contain text, other elements, or be empty. Examples of empty HTML elements are "hr", "br" and "img".*

*Examples:*

*<body>some text</body>  
  
<message>some text</message>*

*Attributes*

*Attributes provide****extra information about elements****.*

*Attributes are always placed inside the opening tag of an element. Attributes always come in name/value pairs. The following "img" element has additional information about a source file:*

*<img src="computer.gif" />*

*The name of the element is "img". The name of the attribute is "src". The value of the attribute is "computer.gif". Since the element itself is empty it is closed by a " /".*

*Entities*

*Some characters have a special meaning in XML, like the less than sign (<) that defines the start of an XML tag.*

*Most of you know the HTML entity: "&nbsp;". This "no-breaking-space" entity is used in HTML to insert an extra space in a document. Entities are expanded when a document is parsed by an XML parser.*

*The following entities are predefined in XML:*

|  |  |
| --- | --- |
| ***Entity References*** | ***Character*** |
| *&lt;* | *<* |
| *&gt;* | *>* |
| *&amp;* | *&* |
| *&quot;* | *"* |
| *&apos;* | *'* |

*PCDATA*

*PCDATA means parsed character data.*

*Think of character data as the text found between the start tag and the end tag of an XML element.*

***PCDATA is text that WILL be parsed by a parser****.****The text will be examined by the parser for entities and markup****. //**дзве асноўныя задачы працэса парсінга**//*

*Tags inside the text will be treated as markup and entities will be expanded.*

*However, parsed character data should not contain any &, <, or > characters; these need to be represented by the &amp; &lt; and &gt; entities, respectively.*

*CDATA*

*CDATA means character data.*

***CDATA is text that will NOT be parsed by a parser****. Tags inside the text will NOT be treated as markup and entities will not be expanded.*

[*https://www.w3schools.com/xml/xml\_dtd\_elements.asp*](https://www.w3schools.com/xml/xml_dtd_elements.asp)

*DTD - Elements*

*In a DTD, elements are declared with an ELEMENT declaration.*

*Declaring Elements*

*In a DTD, XML elements are declared with the following syntax:*

*<!ELEMENT element-name category>  
or  
<!ELEMENT element-name (element-content)>*

*Empty Elements*

*Empty elements are declared with the category keyword EMPTY:*

*<!ELEMENT element-name EMPTY>  
  
Example:  
  
<!ELEMENT br EMPTY>  
  
XML example:  
  
<br />*

*Elements with Parsed Character Data*

*Elements with only parsed character data are declared with #PCDATA inside parentheses:*

*<!ELEMENT element-name (#PCDATA)>  
  
Example:  
  
<!ELEMENT from (#PCDATA)>*

*Elements with any Contents*

*Elements declared with the category keyword ANY, can contain any combination of parsable data:*

*<!ELEMENT element-name ANY>  
  
Example:  
  
<!ELEMENT note ANY>*

*Elements with Children (sequences)*

*Elements with one or more children are declared with the name of the children elements inside parentheses:*

*<!ELEMENT element-name (child1)>  
or  
<!ELEMENT element-name (child1,child2,...)>  
  
Example:  
  
<!ELEMENT note (to,from,heading,body)>*

*When children are declared in a sequence separated by commas, the children must appear in the same sequence in the document. In a full declaration, the children must also be declared, and the children can also have children. The full declaration of the "note" element is:*

*<!ELEMENT note (to,from,heading,body)>  
<!ELEMENT to (#PCDATA)>  
<!ELEMENT from (#PCDATA)>  
<!ELEMENT heading (#PCDATA)>  
<!ELEMENT body (#PCDATA)>*

*Declaring Only One Occurrence of an Element*

*<!ELEMENT element-name (child-name)>  
  
Example:  
  
<!ELEMENT note (message)>*

*The example above declares that the child element "message" must occur once, and only once inside the "note" element.*

*Declaring Minimum One Occurrence of an Element*

*<!ELEMENT element-name (child-name+)>  
  
Example:  
  
<!ELEMENT note (message+)>*

*The + sign in the example above declares that the child element "message" must occur one or more times inside the "note" element.*

*Declaring Zero or More Occurrences of an Element*

*<!ELEMENT element-name (child-name\*)>  
  
Example:  
  
<!ELEMENT note (message\*)>*

*The \* sign in the example above declares that the child element "message" can occur zero or more times inside the "note" element.*

*Declaring Zero or One Occurrences of an Element*

*<!ELEMENT element-name (child-name?)>  
  
Example:  
  
<!ELEMENT note (message?)>*

*The ? sign in the example above declares that the child element "message" can occur zero or one time inside the "note" element.*

*Declaring either/or Content*

*<!ELEMENT note (to,from,header,(message|body))>*

*The example above declares that the "note" element must contain a "to" element, a "from" element, a "header" element, and either a "message" or a "body" element.*

*Declaring Mixed Content*

*<!ELEMENT note (#PCDATA|to|from|header|message)\*>*

*The example above declares that the "note" element can contain zero or more occurrences of parsed character data, "to", "from", "header", or "message" elements.*

[*https://www.w3schools.com/xml/xml\_dtd\_attributes.asp*](https://www.w3schools.com/xml/xml_dtd_attributes.asp)

*DTD - Attributes*

*In a DTD, attributes are declared with an ATTLIST declaration.*

*Declaring Attributes*

*An attribute declaration has the following syntax:*

*<!ATTLIST element-name attribute-name attribute-type attribute-value>  
  
DTD example:  
  
<!ATTLIST payment type CDATA "check">  
  
XML example:  
  
<payment type="check" />*

*The****attribute-type****can be one of the following:*

|  |  |
| --- | --- |
| ***Type*** | ***Description*** |
| *CDATA* | *The value is character data* |
| *(en1|en2|..)* | *The value must be one from an enumerated list* |
| *ID* | *The value is a unique id* |
| *IDREF* | *The value is the id of another element* |
| *IDREFS* | *The value is a list of other ids* |
| *NMTOKEN* | *The value is a valid XML name* |
| *NMTOKENS* | *The value is a list of valid XML names* |
| *ENTITY* | *The value is an entity* |
| *ENTITIES* | *The value is a list of entities* |
| *NOTATION* | *The value is a name of a notation* |
| *xml:* | *The value is a predefined xml value* |

*The****attribute-value****can be one of the following:*

|  |  |
| --- | --- |
| ***Value*** | ***Explanation*** |
| *value* | *The default value of the attribute* |
| *#REQUIRED* | *The attribute is required* |
| *#IMPLIED* | *The attribute is optional* |
| *#FIXED value* | *The attribute value is fixed* |

*A Default Attribute Value*

*DTD:  
<!ELEMENT square EMPTY>  
<!ATTLIST square width CDATA "0">  
  
Valid XML:  
<square width="100" />*

*In the example above, the "square" element is defined to be an empty element with a "width" attribute of  type CDATA. If no width is specified, it has a default value of 0. // што значыць, што тып атрыбута – гэта CDATA? //*

*#REQUIRED*

*Syntax*

*<!ATTLIST element-name attribute-name attribute-type #REQUIRED>*

*Example*

*DTD:  
<!ATTLIST person number CDATA #REQUIRED>  
  
Valid XML:  
<person number="5677" />  
  
Invalid XML:  
<person />*

*Use the #REQUIRED keyword if you don't have an option for a default value, but still want to force the attribute to be present.*

*#IMPLIED*

*Syntax*

*<!ATTLIST element-name attribute-name attribute-type #IMPLIED>*

*Example*

*DTD:  
<!ATTLIST contact fax CDATA #IMPLIED>  
  
Valid XML:  
<contact fax="555-667788" />  
  
Valid XML:  
<contact />*

*Use the #IMPLIED keyword if you don't want to force the author to include an attribute, and you don't have an option for a default value.*

*#FIXED*

*Syntax*

*<!ATTLIST element-name attribute-name attribute-type #FIXED "value">*

*Example*

*DTD:  
<!ATTLIST sender company CDATA #FIXED "Microsoft">  
  
Valid XML:  
<sender company="Microsoft" />  
  
Invalid XML:  
<sender company="W3Schools" />*

*Use the #FIXED keyword when you want an attribute to have a fixed value without allowing the author to change it. If an author includes another value, the XML parser will return an error.*

*// Хром не папярэдзіў мяне аб тым, што я ўвёў няправільны атрыбут згодна з DTD //*

*Enumerated Attribute Values*

*Syntax*

*<!ATTLIST element-name attribute-name (en1|en2|..) default-value>*

*Example*

*DTD:  
<!ATTLIST payment type (check|cash) "cash">  
  
XML example:  
<payment type="check" />  
or  
<payment type="cash" />*

*Use enumerated attribute values when you want the attribute value to be one of a fixed set of legal values.*

[*https://www.w3schools.com/xml/xml\_dtd\_el\_vs\_attr.asp*](https://www.w3schools.com/xml/xml_dtd_el_vs_attr.asp)

*XML Elements vs. Attributes*

*In XML, there are no rules about when to use attributes, and when to use child elements.*

*Use of Elements vs. Attributes*

*Data can be stored in child elements or in attributes.*

*Take a look at these examples:*

*<person sex="female">  
  <firstname>Anna</firstname>  
  <lastname>Smith</lastname>  
</person>*

*<person>  
  <sex>female</sex>  
  <firstname>Anna</firstname>  
  <lastname>Smith</lastname>  
</person>*

*In the first example sex is an attribute. In the last, sex is a child element. Both examples provide the same information. // цудоўны прыклад рознай рэпрэзентацыі амаль аднаго і таго ж сэнсу. Аспекты трэпр //*

*There are no rules about when to use attributes, and when to use child elements. My experience is that attributes are handy in HTML, but in XML you should try to avoid them. Use child elements if the information feels like data.*

*My Favorite Way*

*// чый гэта любімы спосаб? //*

***I like to store data in child elements.***

*The following three XML documents contain exactly the same information:*

*A date attribute is used in the first example:*

*<note date="12/11/2002">  
  <to>Tove</to>  
  <from>Jani</from>  
  <heading>Reminder</heading>  
  <body>Don't forget me this weekend!</body>  
</note>*

*A date element is used in the second example:*

*<note>  
  <date>12/11/2002</date>  
  <to>Tove</to>  
  <from>Jani</from>  
  <heading>Reminder</heading>  
  <body>Don't forget me this weekend!</body>  
</note>*

*An expanded date element is used in the third: (THIS IS MY FAVORITE):*

*<note>  
  <date>  
    <day>12</day>  
    <month>11</month>  
    <year>2002</year>  
  </date>  
  <to>Tove</to>  
  <from>Jani</from>  
  <heading>Reminder</heading>  
  <body>Don't forget me this weekend!</body>  
</note>*

*Avoid using attributes?*

*Should you avoid using attributes?*

*Some of the problems with attributes are:*

* *attributes cannot contain multiple values (child elements can) // intbu //*
* *attributes are not easily expandable (for future changes) // зразумець выяўней. Чаму? Я разумею, што элементы лёгка пашырыць. Але чаму цяжка пашырыць атрыбуты, я разумею не так добра… //*
* *attributes cannot describe structures (child elements can) // праз адносіны ўкладзенасці //*
* *attributes are more difficult to manipulate by program code*
* *attribute values are not easy to test against a DTD // чаму гэта важна? Гэта ж задача парсера? Ці час парсінга на парадак узрастае? //*

*If you use attributes as containers for data, you end up with documents that are difficult to read and maintain. Try to use****elements****to describe data. Use attributes only to provide information that is not relevant to the data.*

*Don't end up like this (this is not how XML should be used):*

*<note day="12" month="11" year="2002"  
to="Tove" from="Jani" heading="Reminder"  
body="Don't forget me this weekend!">  
</note>*

*An Exception to my Attribute Rule*

*Rules always have exceptions.*

*My rule about attributes has one exception:*

*Sometimes I assign ID references to elements. These ID references can be used to access XML elements in much the same way as the NAME or ID attributes in HTML. This example demonstrates this:*

*<messages>  
<note id="p501">  
  <to>Tove</to>  
  <from>Jani</from>  
  <heading>Reminder</heading>  
  <body>Don't forget me this weekend!</body>  
</note>  
  
<note id="p502">  
  <to>Jani</to>  
  <from>Tove</from>  
  <heading>Re: Reminder</heading>  
  <body>I will not!</body>  
</note>  
</messages>*

*The ID in these examples is just a counter, or a unique identifier, to identify the different notes in the XML file, and not a part of the note data. // я зразумеў асноўную думку аўтара… нешта ў гэтым ёсць… Магчыма, атрыбутамі яшчэ можна пазначаць нейкія канстанты, якія заўжды фіксаваныя ў элементах. Паразважаць… што ёсць заўжды? //*

*What I am trying to say here is that metadata (data about data) should be stored as attributes, and that data itself should be stored as elements. // не ўсё так проста. Звычайна і першае, і другое ёсць часткі структуры сісвед. Паразважаць. //*

[*https://www.w3schools.com/xml/xml\_dtd\_entities.asp*](https://www.w3schools.com/xml/xml_dtd_entities.asp)

*DTD - Entities*

*Entities are used to define shortcuts to special characters.*

*Entities can be declared internal or external.*

*An Internal Entity Declaration*

*Syntax*

*<!ENTITY entity-name "entity-value">*

*Example*

*DTD Example:  
  
<!ENTITY writer "Donald Duck.">  
<!ENTITY copyright "Copyright W3Schools.">  
  
XML example:  
  
<author>&writer;&copyright;</author>*

***Note:****An entity has three parts: an ampersand (&), an entity name, and a semicolon (;).*

*An External Entity Declaration*

*Syntax*

*<!ENTITY entity-name SYSTEM "URI/URL">*

*Example*

*DTD Example:  
  
<!ENTITY writer SYSTEM "https://www.w3schools.com/entities.dtd">  
<!ENTITY copyright SYSTEM "https://www.w3schools.com/entities.dtd">  
  
XML example:  
  
<author>&writer;&copyright;</author>*

[*https://www.w3schools.com/xml/xml\_dtd\_examples.asp*](https://www.w3schools.com/xml/xml_dtd_examples.asp)

*// паказаны прыклады. Пра натацыі нічога не сказана. Відаць, гэта не так і важна… //*

XML Schema

[*https://www.w3schools.com/xml/schema\_intro.asp*](https://www.w3schools.com/xml/schema_intro.asp)

*// пачынаем вывучаць схемы //*

*XML Schema Tutorial*

*What is an XML Schema?*

*An XML Schema describes the structure of an XML document.*

*The XML Schema language is also referred to as XML Schema Definition (XSD).*

*XSD Example*

*<?xml version="1.0"?>  
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">  
  
<xs:element name="note">  
  <xs:complexType>  
    <xs:sequence>  
      <xs:element name="to" type="xs:string"/>  
      <xs:element name="from" type="xs:string"/>  
      <xs:element name="heading" type="xs:string"/>  
      <xs:element name="body" type="xs:string"/>  
    </xs:sequence>  
  </xs:complexType>  
</xs:element>  
  
</xs:schema>*

*The purpose of an XML Schema is to define the legal building blocks of an XML document:*

* *the elements and attributes that can appear in a document*
* *the number of (and order of) child elements*
* *data types for elements and attributes*
* *default and fixed values for elements and attributes*

*Why Learn XML Schema?*

*In the XML world, hundreds of standardized XML formats are in daily use.*

*Many of these XML standards are defined by XML Schemas.*

*XML Schema is an XML-based (and more powerful) alternative to DTD.*

*XML Schemas Support Data Types*

*One of the greatest strength of XML Schemas is the support for data types.*

* *It is easier to describe allowable document content*
* *It is easier to validate the correctness of data*
* *It is easier to define data facets (restrictions on data)*
* *It is easier to define data patterns (data formats)*
* *It is easier to convert data between different data types*

*// не вельмі добра разумею, як суадносяцца між сабой гэтыя катэгорыі //*

*// базавыя задачы галіны аналіза дадзен //*

*XML Schemas use XML Syntax*

*Another great strength about XML Schemas is that they are written in XML.*

* *You don't have to learn a new language*
* *You can use your XML editor to edit your Schema files*
* *You can use your XML parser to parse your Schema files*
* *You can manipulate your Schema with the XML DOM*
* *You can transform your Schema with XSLT*

*// варта лепей зразумець перавагі другога і трэцяга //*

*XML Schemas are extensible, because they are written in XML.*

*With an extensible Schema definition you can:*

* *Reuse your Schema in other Schemas*
* *Create your own data types derived from the standard types*
* *Reference multiple schemas in the same document*

*// моцнае наступства пашыраемасці – можна дадаваць змест блокамі. Будаваць змест блокамі. Модульнасць xml. Другое і трэцяе мне варта лепей зразумець. Чаму гэтага нельга рабіць з DTD? //*

*XML Schemas Secure Data Communication*

*When sending data from a sender to a receiver, it is essential that both parts have the same "expectations" about the content. // fundamental aspect of network technologies //*

*With XML Schemas, the sender can describe the data in a way that the receiver will understand.*

*A date like: "03-11-2004" will, in some countries, be interpreted as 3.November and in other countries as 11.March.*

*However, an XML element with a data type like this:*

*<date type="date">2004-03-11</date>*

*ensures a mutual understanding of the content, because the XML data type "date" requires the format "YYYY-MM-DD".*

*// intbu the given example. How can it possible to limit format of element content? //*

*Well-Formed is Not Enough*

*A well-formed XML document is a document that conforms to the XML syntax rules, like:*

* *it must begin with the XML declaration*
* *it must have one unique root element*
* *start-tags must have matching end-tags*
* *elements are case sensitive*
* *all elements must be closed*
* *all elements must be properly nested*
* *all attribute values must be quoted*
* *entities must be used for special characters*

*Even if documents are well-formed they can still contain errors, and those errors can have serious consequences.*

*Think of the following situation: you order 5 gross of laser printers, instead of 5 laser printers. With XML Schemas, most of these errors can be caught by your validating software.*

*// перавагі xsd //*

[*https://www.w3schools.com/xml/schema\_howto.asp*](https://www.w3schools.com/xml/schema_howto.asp)

*XSD How To?*

*XML documents can have a reference to a DTD or to an XML Schema.*

*A Simple XML Document*

*Look at this simple XML document called "note.xml":*

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

*A DTD File*

*The following example is a DTD file called "note.dtd" that defines the elements of the XML document above ("note.xml"):*

*<!ELEMENT note (to, from, heading, body)>  
<!ELEMENT to (#PCDATA)>  
<!ELEMENT from (#PCDATA)>  
<!ELEMENT heading (#PCDATA)>  
<!ELEMENT body (#PCDATA)>*

*The first line defines the note element to have four child elements: "to, from, heading, body".*

*Line 2-5 defines the to, from, heading, body elements to be of type "#PCDATA".*

*An XML Schema*

*The following example is an XML Schema file called "note.xsd" that defines the elements of the XML document above ("note.xml"):*

*<?xml version="1.0"?>  
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"  
targetNamespace="https://www.w3schools.com"  
xmlns="https://www.w3schools.com"  
elementFormDefault="qualified">  
  
<xs:element name="note">  
  <xs:complexType>  
    <xs:sequence>  
      <xs:element name="to" type="xs:string"/>  
      <xs:element name="from" type="xs:string"/>  
      <xs:element name="heading" type="xs:string"/>  
      <xs:element name="body" type="xs:string"/>  
    </xs:sequence>  
  </xs:complexType>  
</xs:element>  
  
</xs:schema>*

*The note element is a****complex type****because it contains other elements. The other elements (to, from, heading, body) are****simple types****because they do not contain other elements. You will learn more about simple and complex types in the following chapters.*

*A Reference to a DTD*

*This XML document has a reference to a DTD:*

*<?xml version="1.0"?>  
  
<!DOCTYPE note SYSTEM  
"https://www.w3schools.com/xml/note.dtd">  
  
<note>  
  <to>Tove</to>  
  <from>Jani</from>  
  <heading>Reminder</heading>  
  <body>Don't forget me this weekend!</body>  
</note>*

*A Reference to an XML Schema*

*This XML document has a reference to an XML Schema:*

*<?xml version="1.0"?>  
  
<note  
xmlns="https://www.w3schools.com"  
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
xsi:schemaLocation="https://www.w3schools.com note.xsd">  
  <to>Tove</to>  
  <from>Jani</from>  
  <heading>Reminder</heading>  
  <body>Don't forget me this weekend!</body>  
</note>*

[*https://www.w3schools.com/xml/schema\_schema.asp*](https://www.w3schools.com/xml/schema_schema.asp)

*XSD - The <schema> Element*

*The <schema> element is the root element of every XML Schema.*

*The <schema> Element*

*The <schema> element is the root element of every XML Schema:*

*<?xml version="1.0"?>  
  
<xs:schema>  
...  
...  
</xs:schema>*

*The <schema> element may contain some attributes. A schema declaration often looks something like this:*

*<?xml version="1.0"?>  
  
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"  
targetNamespace="https://www.w3schools.com"  
xmlns="https://www.w3schools.com"  
elementFormDefault="qualified">  
...  
...  
</xs:schema>*

*The following fragment:*

*xmlns:xs="http://www.w3.org/2001/XMLSchema"*

*indicates that the elements and data types used in the schema come from the "http://www.w3.org/2001/XMLSchema" namespace. It also specifies that the elements and data types that come from the "http://www.w3.org/2001/XMLSchema" namespace should be prefixed with****xs:***

*This fragment:*

*targetNamespace="https://www.w3schools.com"*

*indicates that the elements defined by this schema (note, to, from, heading, body.) come from the "https://www.w3schools.com" namespace.*

*This fragment:*

*xmlns="https://www.w3schools.com"*

*indicates that the default namespace is "https://www.w3schools.com".*

*This fragment:*

*elementFormDefault="qualified"*

*indicates that any elements used by the XML instance document which were declared in this schema must be namespace qualified.*

*Referencing a Schema in an XML Document*

*This XML document has a reference to an XML Schema:*

*<?xml version="1.0"?>  
  
<note xmlns="https://www.w3schools.com"  
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
xsi:schemaLocation="https://www.w3schools.com note.xsd">  
  
<to>Tove</to>  
<from>Jani</from>  
<heading>Reminder</heading>  
<body>Don't forget me this weekend!</body>  
</note>*

*The following fragment:*

*xmlns="https://www.w3schools.com"*

*specifies the default namespace declaration. This declaration tells the schema-validator that all the elements used in this XML document are declared in the "https://www.w3schools.com" namespace.*

*Once you have the XML Schema Instance namespace available:*

*xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"*

*you can use the schemaLocation attribute. This attribute has two values, separated by a space. The first value is the namespace to use. The second value is the location of the XML schema to use for that namespace:*

*xsi:schemaLocation="https://www.w3schools.com note.xsd"*

[*https://www.w3schools.com/xml/schema\_simple.asp*](https://www.w3schools.com/xml/schema_simple.asp)

*XSD Simple Elements*

*XML Schemas define the elements of your XML files.*

*A simple element is an XML element that contains only text. It cannot contain any other elements or attributes.*

*However, the "only text" restriction is quite misleading. The text can be of many different types. It can be one of the types included in the XML Schema definition (boolean, string, date, etc.), or it can be a custom type that you can define yourself.*

*// лепей зразумець гэту магчымасць схемаў //*

*You can also add restrictions (facets) to a data type in order to limit its content, or you can require the data to match a specific pattern.*

*Defining a Simple Element*

*The syntax for defining a simple element is:*

*<xs:element name="xxx" type="yyy"/>*

*where xxx is the name of the element and yyy is the data type of the element.*

*XML Schema has a lot of built-in data types. The most common types are:*

* *xs:string*
* *xs:decimal*
* *xs:integer*
* *xs:boolean*
* *xs:date*
* *xs:time*

*Example*

*Here are some XML elements:*

*<lastname>Refsnes</lastname>  
<age>36</age>  
<dateborn>1970-03-27</dateborn>*

*And here are the corresponding simple element definitions:*

*<xs:element name="lastname" type="xs:string"/>  
<xs:element name="age" type="xs:integer"/>  
<xs:element name="dateborn" type="xs:date"/>*

*Default and Fixed Values for Simple Elements*

*Simple elements may have a default value OR a fixed value specified.*

*A default value is automatically assigned to the element when no other value is specified.*

*In the following example the default value is "red":*

*<xs:element name="color" type="xs:string" default="red"/>*

*A fixed value is also automatically assigned to the element, and you cannot specify another value.*

*In the following example the fixed value is "red":*

*<xs:element name="color" type="xs:string" fixed="red"/>*

[*https://www.w3schools.com/xml/schema\_simple\_attributes.asp*](https://www.w3schools.com/xml/schema_simple_attributes.asp)

*XSD Attributes*

*All attributes are declared as simple types.*

*What is an Attribute?*

*Simple elements cannot have attributes. If an element has attributes, it is considered to be of a complex type. But the attribute itself is always declared as a simple type.*

*How to Define an Attribute?*

*The syntax for defining an attribute is:*

*<xs:attribute name="xxx" type="yyy"/>*

*where xxx is the name of the attribute and yyy specifies the data type of the attribute.*

*XML Schema has a lot of built-in data types. The most common types are:*

* *xs:string*
* *xs:decimal*
* *xs:integer*
* *xs:boolean*
* *xs:date*
* *xs:time*

*Example*

*Here is an XML element with an attribute:*

*<lastname lang="EN">Smith</lastname>*

*And here is the corresponding attribute definition:*

*<xs:attribute name="lang" type="xs:string"/>*

*Default and Fixed Values for Attributes*

*Attributes may have a default value OR a fixed value specified.*

*A default value is automatically assigned to the attribute when no other value is specified.*

*In the following example the default value is "EN":*

*<xs:attribute name="lang" type="xs:string" default="EN"/>*

*A fixed value is also automatically assigned to the attribute, and you cannot specify another value.*

*In the following example the fixed value is "EN":*

*<xs:attribute name="lang" type="xs:string" fixed="EN"/>*

*Optional and Required Attributes*

*Attributes are optional by default. To specify that the attribute is required, use the "use" attribute:*

*<xs:attribute name="lang" type="xs:string" use="required"/>*

*Restrictions on Content*

*When an XML element or attribute has a data type defined, it puts restrictions on the element's or attribute's content.*

*If an XML element is of type "xs:date" and contains a string like "Hello World", the element will not validate.*

*With XML Schemas, you can also add your own restrictions to your XML elements and attributes. These restrictions are called facets. You can read more about facets in the next chapter.*

[*https://www.w3schools.com/xml/schema\_facets.asp*](https://www.w3schools.com/xml/schema_facets.asp)

*XSD Restrictions/Facets*

*Restrictions are used to define acceptable values for XML elements or attributes. Restrictions on XML elements are called facets.*

*Restrictions on Values*

*The following example defines an element called "age" with a restriction. The value of age cannot be lower than 0 or greater than 120:*

*<xs:element name="age">  
  <xs:simpleType>  
    <xs:restriction base="xs:integer">  
      <xs:minInclusive value="0"/>  
      <xs:maxInclusive value="120"/>  
    </xs:restriction>  
  </xs:simpleType>  
</xs:element>*

*Restrictions on a Set of Values*

*To limit the content of an XML element to a set of acceptable values, we would use the enumeration constraint.*

*The example below defines an element called "car" with a restriction. The only acceptable values are: Audi, Golf, BMW:*

*<xs:element name="car">  
  <xs:simpleType>  
    <xs:restriction base="xs:string">  
      <xs:enumeration value="Audi"/>  
      <xs:enumeration value="Golf"/>  
      <xs:enumeration value="BMW"/>  
    </xs:restriction>  
  </xs:simpleType>  
</xs:element>*

*The example above could also have been written like this:*

*<xs:element name="car" type="carType"/>  
  
<xs:simpleType name="carType">  
  <xs:restriction base="xs:string">  
    <xs:enumeration value="Audi"/>  
    <xs:enumeration value="Golf"/>  
    <xs:enumeration value="BMW"/>  
  </xs:restriction>  
</xs:simpleType>*

*// вось прыклад модульнасці. DTD проста не прадастаўляе магчымасці вызначыць асобна тып элемента і выкарыстоўваць яго дзе-нідзе. Паразважаць. Думаю, тэхнічна гэта можна, але кагнетыўна не так проста. Там ёсць толькі ключавыя словы ELEMENT – і проста будзе кагнетыўна цяжка кіраваць тыпамі дадзен. //*

***Note:****In this case the type "carType" can be used by other elements because it is not a part of the "car" element.*

*// прыклад вызначэнне тыпу дадзен? //*

*Restrictions on a Series of Values*

*To limit the content of an XML element to define a series of numbers or letters that can be used, we would use the pattern constraint.*

*The example below defines an element called "letter" with a restriction. The only acceptable value is ONE of the LOWERCASE letters from a to z:*

*<xs:element name="letter">  
  <xs:simpleType>  
    <xs:restriction base="xs:string">  
      <xs:pattern value="[a-z]"/>  
    </xs:restriction>  
  </xs:simpleType>  
</xs:element>*

*The next example defines an element called "initials" with a restriction. The only acceptable value is THREE of the UPPERCASE letters from a to z:*

*<xs:element name="initials">  
  <xs:simpleType>  
    <xs:restriction base="xs:string">  
      <xs:pattern value="[A-Z][A-Z][A-Z]"/>  
    </xs:restriction>  
  </xs:simpleType>  
</xs:element>*

*The next example also defines an element called "initials" with a restriction. The only acceptable value is THREE of the LOWERCASE OR UPPERCASE letters from a to z:*

*<xs:element name="initials">  
  <xs:simpleType>  
    <xs:restriction base="xs:string">  
      <xs:pattern value="[a-zA-Z][a-zA-Z][a-zA-Z]"/>  
    </xs:restriction>  
  </xs:simpleType>  
</xs:element>*

*The next example defines an element called "choice" with a restriction. The only acceptable value is ONE of the following letters: x, y, OR z:*

*<xs:element name="choice">  
  <xs:simpleType>  
    <xs:restriction base="xs:string">  
      <xs:pattern value="[xyz]"/>  
    </xs:restriction>  
  </xs:simpleType>  
</xs:element>*

*The next example defines an element called "prodid" with a restriction. The only acceptable value is FIVE digits in a sequence, and each digit must be in a range from 0 to 9:*

*<xs:element name="prodid">  
  <xs:simpleType>  
    <xs:restriction base="xs:integer">  
      <xs:pattern value="[0-9][0-9][0-9][0-9][0-9]"/>  
    </xs:restriction>  
  </xs:simpleType>  
</xs:element>*

*// для задання патэрнаў выкарыстоўваюць рэгулярныя выразы //*

*Other Restrictions on a Series of Values*

*The example below defines an element called "letter" with a restriction. The acceptable value is zero or more occurrences of lowercase letters from a to z:*

*<xs:element name="letter">  
  <xs:simpleType>  
    <xs:restriction base="xs:string">  
      <xs:pattern value="([a-z])\*"/>  
    </xs:restriction>  
  </xs:simpleType>  
</xs:element>*

*The next example also defines an element called "letter" with a restriction. The acceptable value is one or more pairs of letters, each pair consisting of a lower case letter followed by an upper case letter. For example, "sToP" will be validated by this pattern, but not "Stop" or "STOP" or "stop":*

*<xs:element name="letter">  
  <xs:simpleType>  
    <xs:restriction base="xs:string">  
      <xs:pattern value="([a-z][A-Z])+"/>  
    </xs:restriction>  
  </xs:simpleType>  
</xs:element>*

*The next example defines an element called "gender" with a restriction. The only acceptable value is male OR female:*

*<xs:element name="gender">  
  <xs:simpleType>  
    <xs:restriction base="xs:string">  
      <xs:pattern value="male|female"/>  
    </xs:restriction>  
  </xs:simpleType>  
</xs:element>*

*The next example defines an element called "password" with a restriction. There must be exactly eight characters in a row and those characters must be lowercase or uppercase letters from a to z, or a number from 0 to 9:*

*<xs:element name="password">  
  <xs:simpleType>  
    <xs:restriction base="xs:string">  
      <xs:pattern value="[a-zA-Z0-9]{8}"/>  
    </xs:restriction>  
  </xs:simpleType>  
</xs:element>*

*Restrictions on Whitespace Characters*

*To specify how whitespace characters should be handled, we would use the whiteSpace constraint.*

*This example defines an element called "address" with a restriction. The whiteSpace constraint is set to "preserve", which means that the XML processor WILL NOT remove any white space characters:*

*<xs:element name="address">  
  <xs:simpleType>  
    <xs:restriction base="xs:string">  
      <xs:whiteSpace value="preserve"/>  
    </xs:restriction>  
  </xs:simpleType>  
</xs:element>*

*This example also defines an element called "address" with a restriction. The whiteSpace constraint is set to "replace", which means that the XML processor WILL REPLACE all white space characters (line feeds, tabs, spaces, and carriage returns) with spaces:*

*<xs:element name="address">  
  <xs:simpleType>  
    <xs:restriction base="xs:string">  
      <xs:whiteSpace value="replace"/>  
    </xs:restriction>  
  </xs:simpleType>  
</xs:element>*

*This example also defines an element called "address" with a restriction. The whiteSpace constraint is set to "collapse", which means that the XML processor WILL REMOVE all white space characters (line feeds, tabs, spaces, carriage returns are replaced with spaces, leading and trailing spaces are removed, and multiple spaces are reduced to a single space):*

*<xs:element name="address">  
  <xs:simpleType>  
    <xs:restriction base="xs:string">  
      <xs:whiteSpace value="collapse"/>  
    </xs:restriction>  
  </xs:simpleType>  
</xs:element>*

*Restrictions on Length*

*To limit the length of a value in an element, we would use the length, maxLength, and minLength constraints.*

*This example defines an element called "password" with a restriction. The value must be exactly eight characters:*

*<xs:element name="password">  
  <xs:simpleType>  
    <xs:restriction base="xs:string">  
      <xs:length value="8"/>  
    </xs:restriction>  
  </xs:simpleType>  
</xs:element>*

*This example defines another element called "password" with a restriction. The value must be minimum five characters and maximum eight characters:*

*<xs:element name="password">  
  <xs:simpleType>  
    <xs:restriction base="xs:string">  
      <xs:minLength value="5"/>  
      <xs:maxLength value="8"/>  
    </xs:restriction>  
  </xs:simpleType>  
</xs:element>*

*Restrictions for Datatypes*

|  |  |
| --- | --- |
| ***Constraint*** | ***Description*** |
| *enumeration* | *Defines a list of acceptable values* |
| *fractionDigits* | *Specifies the maximum number of decimal places allowed. Must be equal to or greater than zero* |
| *length* | *Specifies the exact number of characters or list items allowed. Must be equal to or greater than zero* |
| *maxExclusive* | *Specifies the upper bounds for numeric values (the value must be less than this value)* |
| *maxInclusive* | *Specifies the upper bounds for numeric values (the value must be less than or equal to this value)* |
| *maxLength* | *Specifies the maximum number of characters or list items allowed. Must be equal to or greater than zero* |
| *minExclusive* | *Specifies the lower bounds for numeric values (the value must be greater than this value)* |
| *minInclusive* | *Specifies the lower bounds for numeric values (the value must be greater than or equal to this value)* |
| *minLength* | *Specifies the minimum number of characters or list items allowed. Must be equal to or greater than zero* |
| *pattern* | *Defines the exact sequence of characters that are acceptable* |
| *totalDigits* | *Specifies the exact number of digits allowed. Must be greater than zero* |
| *whiteSpace* | *Specifies how white space (line feeds, tabs, spaces, and carriage returns) is handled* |

[*https://www.w3schools.com/xml/schema\_complex.asp*](https://www.w3schools.com/xml/schema_complex.asp)

*XSD Complex Elements*

*A complex element contains other elements and/or attributes.*

*What is a Complex Element?*

*A complex element is an XML element that contains other elements and/or attributes.*

*There are four kinds of complex elements:*

* *empty elements*
* *elements that contain only other elements*
* *elements that contain only text*
* *elements that contain both other elements and text*

***Note:****Each of these elements may contain attributes as well!*

*Examples of Complex Elements*

*A complex XML element, "product", which is empty:*

*<product pid="1345"/>*

*A complex XML element, "employee", which contains only other elements:*

*<employee>  
  <firstname>John</firstname>  
  <lastname>Smith</lastname>  
</employee>*

*A complex XML element, "food", which contains only text:*

*<food type="dessert">Ice cream</food>*

*A complex XML element, "description", which contains both elements and text:*

*<description>  
It happened on <date lang="norwegian">03.03.99</date> ....  
</description>*

*// аспекты маёй сістэмы. паразважаць аб прыгоднасці xml для стварэння метам //*

*How to Define a Complex Element*

*Look at this complex XML element, "employee", which contains only other elements:*

*<employee>  
  <firstname>John</firstname>  
  <lastname>Smith</lastname>  
</employee>*

*We can define a complex element in an XML Schema two different ways:*

*1. The "employee" element can be declared directly by naming the element, like this:*

*<xs:element name="employee">  
  <xs:complexType>  
    <xs:sequence>  
      <xs:element name="firstname" type="xs:string"/>  
      <xs:element name="lastname" type="xs:string"/>  
    </xs:sequence>  
  </xs:complexType>  
</xs:element>*

*If you use the method described above, only the "employee" element can use the specified complex type. Note that the child elements, "firstname" and "lastname", are surrounded by the <sequence> indicator. This means that the child elements must appear in the same order as they are declared. You will learn more about indicators in the XSD Indicators chapter.*

*2. The "employee" element can have a type attribute that refers to the name of the complex type to use:*

*<xs:element name="employee" type="personinfo"/>  
  
<xs:complexType name="personinfo">  
  <xs:sequence>  
    <xs:element name="firstname" type="xs:string"/>  
    <xs:element name="lastname" type="xs:string"/>  
  </xs:sequence>  
</xs:complexType>*

*// а вось і механізмы стварэння тыпаў у xml //*

*If you use the method described above, several elements can refer to the same complex type, like this:*

*<xs:element name="employee" type="personinfo"/>  
<xs:element name="student" type="personinfo"/>  
<xs:element name="member" type="personinfo"/>  
  
<xs:complexType name="personinfo">  
  <xs:sequence>  
    <xs:element name="firstname" type="xs:string"/>  
    <xs:element name="lastname" type="xs:string"/>  
  </xs:sequence>  
</xs:complexType>*

*You can also base a complex element on an existing complex element and add some elements, like this:*

*<xs:element name="employee" type="fullpersoninfo"/>  
  
<xs:complexType name="personinfo">  
  <xs:sequence>  
    <xs:element name="firstname" type="xs:string"/>  
    <xs:element name="lastname" type="xs:string"/>  
  </xs:sequence>  
</xs:complexType>  
  
<xs:complexType name="fullpersoninfo">  
  <xs:complexContent>  
    <xs:extension base="personinfo">  
      <xs:sequence>  
        <xs:element name="address" type="xs:string"/>  
        <xs:element name="city" type="xs:string"/>  
        <xs:element name="country" type="xs:string"/>  
      </xs:sequence>  
    </xs:extension>  
  </xs:complexContent>  
</xs:complexType>*

[*https://www.w3schools.com/xml/schema\_complex\_empty.asp*](https://www.w3schools.com/xml/schema_complex_empty.asp)

*XSD Empty Elements*

*An empty complex element cannot have contents, only attributes.*

*Complex Empty Elements*

*An empty XML element:*

*<product prodid="1345" />*

*The "product" element above has no content at all. To define a type with no content, we must define a type that allows elements in its content, but we do not actually declare any elements, like this:*

*<xs:element name="product">  
  <xs:complexType>  
    <xs:complexContent>  
      <xs:restriction base="xs:integer">  
        <xs:attribute name="prodid" type="xs:positiveInteger"/>  
      </xs:restriction>  
    </xs:complexContent>  
  </xs:complexType>  
</xs:element>*

*In the example above, we define a complex type with a complex content. The complexContent element signals that we intend to restrict or extend the content model of a complex type, and the restriction of integer declares one attribute but does not introduce any element content.*

*// intbu complex content attribute //*

*However, it is possible to declare the "product" element more compactly, like this:*

*<xs:element name="product">  
  <xs:complexType>  
    <xs:attribute name="prodid" type="xs:positiveInteger"/>  
  </xs:complexType>  
</xs:element>*

*Or you can give the complexType element a name, and let the "product" element have a type attribute that refers to the name of the complexType (if you use this method, several elements can refer to the same complex type):*

*<xs:element name="product" type="prodtype"/>  
  
<xs:complexType name="prodtype">  
  <xs:attribute name="prodid" type="xs:positiveInteger"/>  
</xs:complexType>*

*// моцная перавага ўвядзення тыпаў дадзен //*

[*https://www.w3schools.com/xml/schema\_complex\_elements.asp*](https://www.w3schools.com/xml/schema_complex_elements.asp)

*XSD Elements Only*

*An "elements-only" complex type contains an element that contains only other elements.*

*Complex Types Containing Elements Only*

*An XML element, "person", that contains only other elements:*

*<person>  
  <firstname>John</firstname>  
  <lastname>Smith</lastname>  
</person>*

*You can define the "person" element in a schema, like this:*

*<xs:element name="person">  
  <xs:complexType>  
    <xs:sequence>  
      <xs:element name="firstname" type="xs:string"/>  
      <xs:element name="lastname" type="xs:string"/>  
    </xs:sequence>  
  </xs:complexType>  
</xs:element>*

*Notice the <xs:sequence> tag. It means that the elements defined ("firstname" and "lastname") must appear in that order inside a "person" element.*

*Or you can give the complexType element a name, and let the "person" element have a type attribute that refers to the name of the complexType (if you use this method, several elements can refer to the same complex type):*

*<xs:element name="person" type="persontype"/>  
  
<xs:complexType name="persontype">  
  <xs:sequence>  
    <xs:element name="firstname" type="xs:string"/>  
    <xs:element name="lastname" type="xs:string"/>  
  </xs:sequence>  
</xs:complexType>*

[*https://www.w3schools.com/xml/schema\_complex\_text.asp*](https://www.w3schools.com/xml/schema_complex_text.asp)

*XSD Text-Only Elements*

*A complex text-only element can contain text and attributes.*

*Complex Text-Only Elements*

*This type contains only simple content (text and attributes), therefore we add a simpleContent element around the content. When using simple content, you must define an extension OR a restriction within the simpleContent element, like this:*

*<xs:element name="somename">  
  <xs:complexType>  
    <xs:simpleContent>  
      <xs:extension base="basetype">  
        ....  
        ....  
      </xs:extension>  
    </xs:simpleContent>  
  </xs:complexType>  
</xs:element>  
  
OR  
  
<xs:element name="somename">  
  <xs:complexType>  
    <xs:simpleContent>  
      <xs:restriction base="basetype">  
        ....  
        ....  
      </xs:restriction>  
    </xs:simpleContent>  
  </xs:complexType>  
</xs:element>*

***Tip:****Use the extension/restriction element to expand or to limit the base simple type for the element.*

*// simpleContent указвае, што тып будзе толькі простым. А complexType мы выкарыстоўваем тады, калі тып, які зараз просты, у будучыні можа пашырыцца і стаць ужо комплексным //*

*Here is an example of an XML element, "shoesize", that contains text-only:*

*<shoesize country="france">35</shoesize>*

*The following example declares a complexType, "shoesize". The content is defined as an integer value, and the "shoesize" element also contains an attribute named "country":*

*<xs:element name="shoesize">  
  <xs:complexType>  
    <xs:simpleContent>  
      <xs:extension base="xs:integer">  
        <xs:attribute name="country" type="xs:string" />  
      </xs:extension>  
    </xs:simpleContent>  
  </xs:complexType>  
</xs:element>*

*We could also give the complexType element a name, and let the "shoesize" element have a type attribute that refers to the name of the complexType (if you use this method, several elements can refer to the same complex type):*

*<xs:element name="shoesize" type="shoetype"/>  
  
<xs:complexType name="shoetype">  
  <xs:simpleContent>  
    <xs:extension base="xs:integer">  
      <xs:attribute name="country" type="xs:string" />  
    </xs:extension>  
  </xs:simpleContent>  
</xs:complexType>*

[*https://www.w3schools.com/xml/schema\_complex\_mixed.asp*](https://www.w3schools.com/xml/schema_complex_mixed.asp)

*XSD Mixed Content*

*A mixed complex type element can contain attributes, elements, and text.*

*Complex Types with Mixed Content*

*An XML element, "letter", that contains both text and other elements:*

*<letter>  
  Dear Mr.<name>John Smith</name>.  
  Your order <orderid>1032</orderid>  
  will be shipped on <shipdate>2001-07-13</shipdate>.  
</letter>*

*The following schema declares the "letter" element:*

*<xs:element name="letter">  
  <xs:complexType mixed="true">  
    <xs:sequence>  
      <xs:element name="name" type="xs:string"/>  
      <xs:element name="orderid" type="xs:positiveInteger"/>  
      <xs:element name="shipdate" type="xs:date"/>  
    </xs:sequence>  
  </xs:complexType>  
</xs:element>*

***Note:****To enable character data to appear between the child-elements of "letter", the mixed attribute must be set to "true". The <xs:sequence> tag means that the elements defined (name, orderid and shipdate) must appear in that order inside a "letter" element.*

*We could also give the complexType element a name, and let the "letter" element have a type attribute that refers to the name of the complexType (if you use this method, several elements can refer to the same complex type):*

*<xs:element name="letter" type="lettertype"/>  
  
<xs:complexType name="lettertype" mixed="true">  
  <xs:sequence>  
    <xs:element name="name" type="xs:string"/>  
    <xs:element name="orderid" type="xs:positiveInteger"/>  
    <xs:element name="shipdate" type="xs:date"/>  
  </xs:sequence>  
</xs:complexType>*

[*https://www.w3schools.com/xml/schema\_complex\_indicators.asp*](https://www.w3schools.com/xml/schema_complex_indicators.asp)

*XSD Indicators*

*We can control HOW elements are to be used in documents with indicators.*

*Indicators*

*There are seven indicators:*

*Order indicators:*

* *All*
* *Choice*
* *Sequence*

*Occurrence indicators:*

* *maxOccurs*
* *minOccurs*

*Group indicators:*

* *Group name*
* *attributeGroup name*

*Order Indicators*

*Order indicators are used to define the order of the elements.*

*All Indicator*

*The <all> indicator specifies that the child elements can appear in any order, and that each child element must occur only once:*

*<xs:element name="person">  
  <xs:complexType>  
    <xs:all>  
      <xs:element name="firstname" type="xs:string"/>  
      <xs:element name="lastname" type="xs:string"/>  
    </xs:all>  
  </xs:complexType>  
</xs:element>*

***Note:****When using the <all> indicator you can set the <minOccurs> indicator to 0 or 1 and the <maxOccurs> indicator can only be set to 1 (the <minOccurs> and <maxOccurs> are described later).*

*Choice Indicator*

*The <choice> indicator specifies that either one child element or another can occur:*

*<xs:element name="person">  
  <xs:complexType>  
    <xs:choice>  
      <xs:element name="employee" type="employee"/>  
      <xs:element name="member" type="member"/>  
    </xs:choice>  
  </xs:complexType>  
</xs:element>*

*Sequence Indicator*

*The <sequence> indicator specifies that the child elements must appear in a specific order:*

*<xs:element name="person">  
   <xs:complexType>  
    <xs:sequence>  
      <xs:element name="firstname" type="xs:string"/>  
      <xs:element name="lastname" type="xs:string"/>  
    </xs:sequence>  
  </xs:complexType>  
</xs:element>*

*// а выкарыстоўваючы іх укладзенасць, мы можам задаваць фармат вельмі гінка. Гэта на парадак мацней за DTD //*

*// у DTD, здаецца, зусім не было магчымасці альтэрнатывы //*

*Occurrence Indicators*

*Occurrence indicators are used to define how often an element can occur.*

***Note:****For all "Order" and "Group" indicators (any, all, choice, sequence, group name, and group reference) the default value for maxOccurs and minOccurs is 1.*

*maxOccurs Indicator*

*The <maxOccurs> indicator specifies the maximum number of times an element can occur:*

*<xs:element name="person">  
  <xs:complexType>  
    <xs:sequence>  
      <xs:element name="full\_name" type="xs:string"/>  
      <xs:element name="child\_name" type="xs:string" maxOccurs="10"/>  
    </xs:sequence>  
  </xs:complexType>  
</xs:element>*

*The example above indicates that the "child\_name" element can occur a minimum of one time (the default value for minOccurs is 1) and a maximum of ten times in the "person" element.*

*minOccurs Indicator*

*The <minOccurs> indicator specifies the minimum number of times an element can occur:*

*<xs:element name="person">  
  <xs:complexType>  
    <xs:sequence>  
      <xs:element name="full\_name" type="xs:string"/>  
      <xs:element name="child\_name" type="xs:string"  
      maxOccurs="10" minOccurs="0"/>  
    </xs:sequence>  
  </xs:complexType>  
</xs:element>*

*The example above indicates that the "child\_name" element can occur a minimum of zero times and a maximum of ten times in the "person" element.*

***Tip:****To allow an element to appear an unlimited number of times, use the maxOccurs="unbounded" statement:*

***A working example:***

*An XML file called "Myfamily.xml":*

*<?xml version="1.0" encoding="UTF-8"?>  
  
<persons xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
xsi:noNamespaceSchemaLocation="family.xsd">  
  
<person>  
  <full\_name>Hege Refsnes</full\_name>  
  <child\_name>Cecilie</child\_name>  
</person>  
  
<person>  
  <full\_name>Tove Refsnes</full\_name>  
  <child\_name>Hege</child\_name>  
  <child\_name>Stale</child\_name>  
  <child\_name>Jim</child\_name>  
  <child\_name>Borge</child\_name>  
</person>  
  
<person>  
  <full\_name>Stale Refsnes</full\_name>  
</person>  
  
</persons>*

*The XML file above contains a root element named "persons". Inside this root element we have defined three "person" elements. Each "person" element must contain a "full\_name" element and it can contain up to five "child\_name" elements.*

*Here is the schema file "family.xsd":*

*<?xml version="1.0" encoding="UTF-8"?>  
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"  
elementFormDefault="qualified">  
  
<xs:element name="persons">  
  <xs:complexType>  
    <xs:sequence>  
      <xs:element name="person" maxOccurs="unbounded">  
        <xs:complexType>  
          <xs:sequence>  
            <xs:element name="full\_name" type="xs:string"/>  
            <xs:element name="child\_name" type="xs:string"  
            minOccurs="0" maxOccurs="5"/>  
          </xs:sequence>  
        </xs:complexType>  
      </xs:element>  
    </xs:sequence>  
  </xs:complexType>  
</xs:element>  
  
</xs:schema>*

*Group Indicators*

*Group indicators are used to define related sets of elements.*

*Element Groups*

*Element groups are defined with the group declaration, like this:*

*<xs:group name="groupname">  
...  
</xs:group>*

*You must define an all, choice, or sequence element inside the group declaration. The following example defines a group named "persongroup", that defines a group of elements that must occur in an exact sequence:*

*<xs:group name="persongroup">  
  <xs:sequence>  
    <xs:element name="firstname" type="xs:string"/>  
    <xs:element name="lastname" type="xs:string"/>  
    <xs:element name="birthday" type="xs:date"/>  
  </xs:sequence>  
</xs:group>*

*After you have defined a group, you can reference it in another definition, like this:*

*<xs:group name="persongroup">  
  <xs:sequence>  
    <xs:element name="firstname" type="xs:string"/>  
    <xs:element name="lastname" type="xs:string"/>  
    <xs:element name="birthday" type="xs:date"/>  
  </xs:sequence>  
</xs:group>  
  
<xs:element name="person" type="personinfo"/>  
  
<xs:complexType name="personinfo">  
  <xs:sequence>  
    <xs:group ref="persongroup"/>  
    <xs:element name="country" type="xs:string"/>  
  </xs:sequence>  
</xs:complexType>*

*// Чым адрозніваюцца групы ад комлексных тыпаў? //*

*// гэта разнавіднасць комплексных тыпаў? //*

*Attribute Groups*

*Attribute groups are defined with the attributeGroup declaration, like this:*

*<xs:attributeGroup name="groupname">  
...  
</xs:attributeGroup>*

*The following example defines an attribute group named "personattrgroup":*

*<xs:attributeGroup name="personattrgroup">  
  <xs:attribute name="firstname" type="xs:string"/>  
  <xs:attribute name="lastname" type="xs:string"/>  
  <xs:attribute name="birthday" type="xs:date"/>  
</xs:attributeGroup>*

*After you have defined an attribute group, you can reference it in another definition, like this:*

*<xs:attributeGroup name="personattrgroup">  
  <xs:attribute name="firstname" type="xs:string"/>  
  <xs:attribute name="lastname" type="xs:string"/>  
  <xs:attribute name="birthday" type="xs:date"/>  
</xs:attributeGroup>  
  
<xs:element name="person">  
  <xs:complexType>  
    <xs:attributeGroup ref="personattrgroup"/>  
  </xs:complexType>  
</xs:element>*

[*https://www.w3schools.com/xml/schema\_complex\_any.asp*](https://www.w3schools.com/xml/schema_complex_any.asp)

*XSD The <any> Element*

*The <any> element enables us to extend the XML document with elements not specified by the schema!*

*The following example is a fragment from an XML schema called "family.xsd". It shows a declaration for the "person" element. By using the <any> element we can extend (after <lastname>) the content of "person" with any element:*

*<xs:element name="person">  
  <xs:complexType>  
    <xs:sequence>  
      <xs:element name="firstname" type="xs:string"/>  
      <xs:element name="lastname" type="xs:string"/>  
      <xs:any minOccurs="0"/>  
    </xs:sequence>  
  </xs:complexType>  
</xs:element>*

*Now we want to extend the "person" element with a "children" element. In this case we can do so, even if the author of the schema above never declared any "children" element.*

*Look at this schema file, called "children.xsd":*

*<?xml version="1.0" encoding="UTF-8"?>  
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"  
targetNamespace="https://www.w3schools.com"  
xmlns="https://www.w3schools.com"  
elementFormDefault="qualified">  
  
<xs:element name="children">  
  <xs:complexType>  
    <xs:sequence>  
      <xs:element name="childname" type="xs:string"  
      maxOccurs="unbounded"/>  
    </xs:sequence>  
  </xs:complexType>  
</xs:element>  
  
</xs:schema>*

*The XML file below (called "Myfamily.xml"), uses components from two different schemas; "family.xsd" and "children.xsd":*

*<?xml version="1.0" encoding="UTF-8"?>  
  
<persons xmlns="http://www.microsoft.com"  
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
xsi:schemaLocation="http://www.microsoft.com family.xsd  
https://www.w3schools.com children.xsd">  
  
<person>  
  <firstname>Hege</firstname>  
  <lastname>Refsnes</lastname>  
  <children>  
    <childname>Cecilie</childname>  
  </children>  
</person>  
  
<person>  
  <firstname>Stale</firstname>  
  <lastname>Refsnes</lastname>  
</person>  
  
</persons>*

*The XML file above is valid because the schema "family.xsd" allows us to extend the "person" element with an optional element after the "lastname" element.*

*The <any> and <anyAttribute> elements are used to make EXTENSIBLE documents! They allow documents to contain additional elements that are not declared in the main XML schema.*

*// моцны сродак пашыраемасці дакументаў. І таксама тут бачна магчымасць спасылацца на некалькі схемаў. Гэта моцна //*

[*https://www.w3schools.com/xml/schema\_complex\_anyattribute.asp*](https://www.w3schools.com/xml/schema_complex_anyattribute.asp)

*XSD The <anyAttribute> Element*

*The <anyAttribute> element enables us to extend the XML document with attributes not specified by the schema!*

*The following example is a fragment from an XML schema called "family.xsd". It shows a declaration for the "person" element. By using the <anyAttribute> element we can add any number of attributes to the "person" element:*

*<xs:element name="person">  
  <xs:complexType>  
    <xs:sequence>  
      <xs:element name="firstname" type="xs:string"/>  
      <xs:element name="lastname" type="xs:string"/>  
    </xs:sequence>  
    <xs:anyAttribute/>  
  </xs:complexType>  
</xs:element>*

*Now we want to extend the "person" element with a "eyecolor" attribute. In this case we can do so, even if the author of the schema above never declared any "eyecolor" attribute.*

*Look at this schema file, called "attribute.xsd":*

*<?xml version="1.0" encoding="UTF-8"?>  
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"  
targetNamespace="https://www.w3schools.com"  
xmlns="https://www.w3schools.com"  
elementFormDefault="qualified">  
  
<xs:attribute name="eyecolor">  
  <xs:simpleType>  
    <xs:restriction base="xs:string">  
      <xs:pattern value="blue|brown|green|grey"/>  
    </xs:restriction>  
  </xs:simpleType>  
</xs:attribute>  
  
</xs:schema>*

*The XML file below (called "Myfamily.xml"), uses components from two different schemas; "family.xsd" and "attribute.xsd":*

*<?xml version="1.0" encoding="UTF-8"?>  
  
<persons xmlns="http://www.microsoft.com"  
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
xsi:SchemaLocation="http://www.microsoft.com family.xsd  
https://www.w3schools.com attribute.xsd">  
  
<person eyecolor="green">  
  <firstname>Hege</firstname>  
  <lastname>Refsnes</lastname>  
</person>  
  
<person eyecolor="blue">  
  <firstname>Stale</firstname>  
  <lastname>Refsnes</lastname>  
</person>  
  
</persons>*

*The XML file above is valid because the schema "family.xsd" allows us to add an attribute to the "person" element.*

*The <any> and <anyAttribute> elements are used to make EXTENSIBLE documents! They allow documents to contain additional elements that are not declared in the main XML schema.*

[*https://www.w3schools.com/xml/schema\_complex\_subst.asp*](https://www.w3schools.com/xml/schema_complex_subst.asp)

*XSD Element Substitution*

*With XML Schemas, one element can substitute another element.*

*Let's say that we have users from two different countries: England and Norway. We would like the ability to let the user choose whether he or she would like to use the Norwegian element names or the English element names in the XML document. // кантэкст выкарыстання дадзенай магчымасці тэхналогіі //*

*To solve this problem, we could define a****substitutionGroup****in the XML schema. First, we declare a head element and then we declare the other elements which state that they are substitutable for the head element.*

*<xs:element name="name" type="xs:string"/>  
<xs:element name="navn" substitutionGroup="name"/>*

*In the example above, the "name" element is the head element and the "navn" element is substitutable for "name".*

*// зразумела, як рабіць інтэрнацыяналізацыю xml дакументаў //*

*Look at this fragment of an XML schema:*

*<xs:element name="name" type="xs:string"/>  
<xs:element name="navn" substitutionGroup="name"/>  
  
<xs:complexType name="custinfo">  
  <xs:sequence>  
    <xs:element ref="name"/>  
  </xs:sequence>  
</xs:complexType>  
  
<xs:element name="customer" type="custinfo"/>  
<xs:element name="kunde" substitutionGroup="customer"/>*

*A valid XML document (according to the schema above) could look like this:*

*<customer>  
  <name>John Smith</name>  
</customer>*

*or like this:*

*<kunde>  
  <navn>John Smith</navn>  
</kunde>*

*// example //*

*Blocking Element Substitution*

*To prevent other elements from substituting with a specified element, use the block attribute:*

*<xs:element name="name" type="xs:string" block="substitution"/>*

*Look at this fragment of an XML schema:*

*<xs:element name="name" type="xs:string" block="substitution"/>  
<xs:element name="navn" substitutionGroup="name"/>  
  
<xs:complexType name="custinfo">  
  <xs:sequence>  
    <xs:element ref="name"/>  
  </xs:sequence>  
</xs:complexType>  
  
<xs:element name="customer" type="custinfo" block="substitution"/>  
<xs:element name="kunde" substitutionGroup="customer"/>*

*A valid XML document (according to the schema above) looks like this:*

*<customer>  
  <name>John Smith</name>  
</customer>*

*BUT THIS IS NO LONGER VALID:*

*<kunde>  
  <navn>John Smith</navn>  
</kunde>*

*Using substitutionGroup*

*The type of the substitutable elements must be the same as, or derived from, the type of the head element. If the type of the substitutable element is the same as the type of the head element you will not have to specify the type of the substitutable element. // можа быць, тут ёсць памылка перакладу? //*

*Note that all elements in the substitutionGroup (the head element and the substitutable elements) must be declared as global elements, otherwise it will not work!*

*What are Global Elements?*

*Global elements are elements that are immediate children of the "schema" element! Local elements are elements nested within other elements.*

*// а гэта ўскладняе задачу. Мы та хочам звычайна ўсе элементы перакладаць. Паразважаць. //*

*// тады трэба глабальна вызначыць усе магчымыя тыпы, а потым праводзіць іх інтэрнацыяналізацыю //*

[*https://www.w3schools.com/xml/schema\_example.asp*](https://www.w3schools.com/xml/schema_example.asp)

*An XSD Example*

*This chapter will demonstrate how to write an XML Schema. You will also learn that a schema can be written in different ways.*

*An XML Document*

*Let's have a look at this XML document called "shiporder.xml":*

*<?xml version="1.0" encoding="UTF-8"?>  
  
<shiporder orderid="889923"  
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
xsi:noNamespaceSchemaLocation="shiporder.xsd">  
  <orderperson>John Smith</orderperson>  
  <shipto>  
    <name>Ola Nordmann</name>  
    <address>Langgt 23</address>  
    <city>4000 Stavanger</city>  
    <country>Norway</country>  
  </shipto>  
  <item>  
    <title>Empire Burlesque</title>  
    <note>Special Edition</note>  
    <quantity>1</quantity>  
    <price>10.90</price>  
  </item>  
  <item>  
    <title>Hide your heart</title>  
    <quantity>1</quantity>  
    <price>9.90</price>  
  </item>  
</shiporder>*

*The XML document above consists of a root element, "shiporder", that contains a required attribute called "orderid". The "shiporder" element contains three different child elements: "orderperson", "shipto" and "item". The "item" element appears twice, and it contains a "title", an optional "note" element, a "quantity", and a "price" element.*

*The line above: xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" tells the XML parser that this document should be validated against a schema. The line: xsi:noNamespaceSchemaLocation="shiporder.xsd" specifies WHERE the schema resides (here it is in the same folder as "shiporder.xml").*

*Create an XML Schema*

*Now we want to create a schema for the XML document above.*

*// спачатку звычайна ствараюць дакумент, а ўжо потым схему. Гэта і ясна, бо так лягчэй. Схема можа каштаваць шмат часу, а ўнёсак зменаў спачатку будзе рэгулярным. Гэта прырода чалавечага разумення. дадзеныя змены і разуменне дакумента xml можна адчуць і без схемы спачатку. Схема – гэта праграмны сродак, які потым дазваляе падтрымліваць зразуметае. Сама па сабе схема можа наводзіць на думкі. Але мне невідавочна, што гэта вельмі-вельмі моцны і важны ўплыў //*

*We start by opening a new file that we will call "shiporder.xsd". To create the schema we could simply follow the structure in the XML document and define each element as we find it. We will start with the standard XML declaration followed by the xs:schema element that defines a schema:*

*<?xml version="1.0" encoding="UTF-8" ?>  
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">  
...  
</xs:schema>*

*In the schema above we use the standard namespace (xs), and the URI associated with this namespace is the Schema language definition, which has the standard value of* [*http://www.w3.org/2001/XMLSchema*](http://www.w3.org/2001/XMLSchema)*.*

*// разнавіднасць стандарта. Даследваць. //*

*Next, we have to define the "shiporder" element. This element has an attribute and it contains other elements, therefore we consider it as a complex type. The child elements of the "shiporder" element is surrounded by a xs:sequence element that defines an ordered sequence of sub elements:*

*<xs:element name="shiporder">  
  <xs:complexType>  
    <xs:sequence>  
      ...  
    </xs:sequence>  
  </xs:complexType>  
</xs:element>*

*Then we have to define the "orderperson" element as a simple type (because it does not contain any attributes or other elements). The type (xs:string) is prefixed with the namespace prefix associated with XML Schema that indicates a predefined schema data type:*

*<xs:element name="orderperson" type="xs:string"/>*

*Next, we have to define two elements that are of the complex type: "shipto" and "item". We start by defining the "shipto" element:*

*<xs:element name="shipto">  
  <xs:complexType>  
    <xs:sequence>  
      <xs:element name="name" type="xs:string"/>  
      <xs:element name="address" type="xs:string"/>  
      <xs:element name="city" type="xs:string"/>  
      <xs:element name="country" type="xs:string"/>  
    </xs:sequence>  
  </xs:complexType>  
</xs:element>*

*With schemas we can define the number of possible occurrences for an element with the maxOccurs and minOccurs attributes. maxOccurs specifies the maximum number of occurrences for an element and minOccurs specifies the minimum number of occurrences for an element. The default value for both maxOccurs and minOccurs is 1!*

*Now we can define the "item" element. This element can appear multiple times inside a "shiporder" element. This is specified by setting the maxOccurs attribute of the "item" element to "unbounded" which means that there can be as many occurrences of the "item" element as the author wishes. Notice that the "note" element is optional. We have specified this by setting the minOccurs attribute to zero:*

*<xs:element name="item" maxOccurs="unbounded">  
  <xs:complexType>  
    <xs:sequence>  
      <xs:element name="title" type="xs:string"/>  
      <xs:element name="note" type="xs:string" minOccurs="0"/>  
      <xs:element name="quantity" type="xs:positiveInteger"/>  
      <xs:element name="price" type="xs:decimal"/>  
    </xs:sequence>  
  </xs:complexType>  
</xs:element>*

*We can now declare the attribute of the "shiporder" element. Since this is a required attribute we specify use="required".*

***Note:****The attribute declarations must always come last:*

*<xs:attribute name="orderid" type="xs:string" use="required"/>*

*Here is the complete listing of the schema file called "shiporder.xsd":*

*<?xml version="1.0" encoding="UTF-8" ?>  
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">  
  
<xs:element name="shiporder">  
  <xs:complexType>  
    <xs:sequence>  
      <xs:element name="orderperson" type="xs:string"/>  
      <xs:element name="shipto">  
        <xs:complexType>  
          <xs:sequence>  
            <xs:element name="name" type="xs:string"/>  
            <xs:element name="address" type="xs:string"/>  
            <xs:element name="city" type="xs:string"/>  
            <xs:element name="country" type="xs:string"/>  
          </xs:sequence>  
        </xs:complexType>  
      </xs:element>  
      <xs:element name="item" maxOccurs="unbounded">  
        <xs:complexType>  
          <xs:sequence>  
            <xs:element name="title" type="xs:string"/>  
            <xs:element name="note" type="xs:string" minOccurs="0"/>  
            <xs:element name="quantity" type="xs:positiveInteger"/>  
            <xs:element name="price" type="xs:decimal"/>  
          </xs:sequence>  
        </xs:complexType>  
      </xs:element>  
    </xs:sequence>  
    <xs:attribute name="orderid" type="xs:string" use="required"/>  
  </xs:complexType>  
</xs:element>  
  
</xs:schema>*

*Divide the Schema*

*The previous design method is very simple, but can be difficult to read and maintain when documents are complex.*

*The next design method is based on defining all elements and attributes first, and then referring to them using the ref attribute.*

*Here is the new design of the schema file ("shiporder.xsd"):*

*<?xml version="1.0" encoding="UTF-8" ?>  
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">  
  
<!-- definition of simple elements -->  
<xs:element name="orderperson" type="xs:string"/>  
<xs:element name="name" type="xs:string"/>  
<xs:element name="address" type="xs:string"/>  
<xs:element name="city" type="xs:string"/>  
<xs:element name="country" type="xs:string"/>  
<xs:element name="title" type="xs:string"/>  
<xs:element name="note" type="xs:string"/>  
<xs:element name="quantity" type="xs:positiveInteger"/>  
<xs:element name="price" type="xs:decimal"/>  
  
<!-- definition of attributes -->  
<xs:attribute name="orderid" type="xs:string"/>  
  
<!-- definition of complex elements -->  
<xs:element name="shipto">  
  <xs:complexType>  
    <xs:sequence>  
      <xs:element ref="name"/>  
      <xs:element ref="address"/>  
      <xs:element ref="city"/>  
      <xs:element ref="country"/>  
    </xs:sequence>  
  </xs:complexType>  
</xs:element>  
  
<xs:element name="item">  
  <xs:complexType>  
    <xs:sequence>  
      <xs:element ref="title"/>  
      <xs:element ref="note" minOccurs="0"/>  
      <xs:element ref="quantity"/>  
      <xs:element ref="price"/>  
    </xs:sequence>  
  </xs:complexType>  
</xs:element>  
  
<xs:element name="shiporder">  
  <xs:complexType>  
    <xs:sequence>  
      <xs:element ref="orderperson"/>  
      <xs:element ref="shipto"/>  
      <xs:element ref="item" maxOccurs="unbounded"/>  
    </xs:sequence>  
    <xs:attribute ref="orderid" use="required"/>  
  </xs:complexType>  
</xs:element>  
  
</xs:schema>*

*// кантэкст выкарыстання атрыбута элемента xs::element пад назвай ref //*

*// але тут сувязі між элементамі не такія відавочныя. Лепей мець і тую схему, і тую. Прычым першую, мне здаецца, можна вывесці з другой. Ці не заўсёды? Паразважаць. //*

*Using Named Types*

*The third design method defines classes or types, that enables us to reuse element definitions. This is done by naming the simpleTypes and complexTypes elements, and then point to them through the type attribute of the element.*

*Here is the third design of the schema file ("shiporder.xsd"):*

*<?xml version="1.0" encoding="UTF-8" ?>  
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">  
  
<xs:simpleType name="stringtype">  
  <xs:restriction base="xs:string"/>  
</xs:simpleType>  
  
<xs:simpleType name="inttype">  
  <xs:restriction base="xs:positiveInteger"/>  
</xs:simpleType>  
  
<xs:simpleType name="dectype">  
  <xs:restriction base="xs:decimal"/>  
</xs:simpleType>  
  
<xs:simpleType name="orderidtype">  
  <xs:restriction base="xs:string">  
    <xs:pattern value="[0-9]{6}"/>  
  </xs:restriction>  
</xs:simpleType>  
  
<xs:complexType name="shiptotype">  
  <xs:sequence>  
    <xs:element name="name" type="stringtype"/>  
    <xs:element name="address" type="stringtype"/>  
    <xs:element name="city" type="stringtype"/>  
    <xs:element name="country" type="stringtype"/>  
  </xs:sequence>  
</xs:complexType>  
  
<xs:complexType name="itemtype">  
  <xs:sequence>  
    <xs:element name="title" type="stringtype"/>  
    <xs:element name="note" type="stringtype" minOccurs="0"/>  
    <xs:element name="quantity" type="inttype"/>  
    <xs:element name="price" type="dectype"/>  
  </xs:sequence>  
</xs:complexType>  
  
<xs:complexType name="shipordertype">  
  <xs:sequence>  
    <xs:element name="orderperson" type="stringtype"/>  
    <xs:element name="shipto" type="shiptotype"/>  
    <xs:element name="item" maxOccurs="unbounded" type="itemtype"/>  
  </xs:sequence>  
  <xs:attribute name="orderid" type="orderidtype" use="required"/>  
</xs:complexType>  
  
<xs:element name="shiporder" type="shipordertype"/>  
  
</xs:schema>*

*The restriction element indicates that the datatype is derived from a W3C XML Schema namespace datatype. So, the following fragment means that the value of the element or attribute must be a string value:*

*<xs:restriction base="xs:string">*

*The restriction element is more often used to apply restrictions to elements. Look at the following lines from the schema above:*

*<xs:simpleType name="orderidtype">  
  <xs:restriction base="xs:string">  
    <xs:pattern value="[0-9]{6}"/>  
  </xs:restriction>  
</xs:simpleType>*

*This indicates that the value of the element or attribute must be a string, it must be exactly six characters in a row, and those characters must be a number from 0 to 9.*

*// потым на практыцы зразумець розніцу між трыма спосабамі. Першы – гэта строгая іерархія xml-дакумента. Другі – гэта калі мы іерархію разбіваем на простыя тыпы і комплексныя, якія ўказваем паасобку, прычым розныя элементы могуць спасылацца адзін на аднаго. Трэці спосаб – гэта калі мы ўводзім тыпы дадзен. І ўжо праз іх апісваем простыя тыпы і комплексныя тыпы, як у другім варыянце. Паразважаць. Прачуць на практыцы. //*

*// у папярэднім метадзе мы вызначалі xs:element элементы, а потым спасылаліся на іх праз атрыбут ref. У дадзеным метадзе мы вызначаем xs:complexType і xs:simpleType і спасылаемся на іх праз атрыбут type. Аднак акрамя ментальнага аспекта (назвы simpleType і complexType больш зразумелая і канкрэтная, чым проста element), чыста з пункту гледжання тэхнічнага аспекта, ці ёсць розніца між апошнімі двума спосабамі? Невідавочна. //*

[*https://www.w3schools.com/xml/schema\_dtypes\_string.asp*](https://www.w3schools.com/xml/schema_dtypes_string.asp)

*XSD String Data Types*

*String data types are used for values that contains character strings.*

*The string data type can contain characters, line feeds, carriage returns, and tab characters.*

*The following is an example of a string declaration in a schema:*

*<xs:element name="customer" type="xs:string"/>*

*An element in your document might look like this:*

*<customer>John Smith</customer>*

*Or it might look like this:*

*<customer>       John Smith     </customer>*

***Note:****The XML processor will not modify the value if you use the string data type.*

*NormalizedString Data Type*

*The normalizedString data type is derived from the String data type.*

*The normalizedString data type also contains characters, but the XML processor will remove line feeds, carriage returns, and tab characters.*

*The following is an example of a normalizedString declaration in a schema:*

*<xs:element name="customer" type="xs:normalizedString"/>*

*An element in your document might look like this:*

*<customer>John Smith</customer>*

*Or it might look like this:*

*<customer>     John Smith     </customer>*

***Note:****In the example above the XML processor will replace the tabs with spaces.*

*Token Data Type*

*The token data type is also derived from the String data type.*

*The token data type also contains characters, but the XML processor will remove line feeds, carriage returns, tabs, leading and trailing spaces, and multiple spaces.*

*The following is an example of a token declaration in a schema:*

*<xs:element name="customer" type="xs:token"/>*

*An element in your document might look like this:*

*<customer>John Smith</customer>*

*Or it might look like this:*

*<customer>     John Smith     </customer>*

***Note:****In the example above the XML processor will remove the tabs.*

*// тоукен – гэта там, дзе няма прабелаў. Дзе ёсць толькі адно слова і ўсё //*

*String Data Types*

*Note that all of the data types below derive from the String data type (except for string itself)!*

|  |  |
| --- | --- |
| ***Name*** | ***Description*** |
| *ENTITIES* |  |
| *ENTITY* |  |
| *ID* | *A string that represents the ID attribute in XML (only used with schema attributes)* |
| *IDREF* | *A string that represents the IDREF attribute in XML (only used with schema attributes)* |
| *IDREFS* |  |
| *language* | *A string that contains a valid language id* |
| *Name* | *A string that contains a valid XML name* |
| *NCName* |  |
| *NMTOKEN* | *A string that represents the NMTOKEN attribute in XML (only used with schema attributes)* |
| *NMTOKENS* |  |
| *normalizedString* | *A string that does not contain line feeds, carriage returns, or tabs* |
| *QName* |  |
| *string* | *A string* |
| *token* | *A string that does not contain line feeds, carriage returns, tabs, leading or trailing spaces, or multiple spaces* |

*Restrictions on String Data Types*

*Restrictions that can be used with String data types:*

* *enumeration*
* *length*
* *maxLength*
* *minLength*
* *pattern (NMTOKENS, IDREFS, and ENTITIES cannot use this constraint)*
* *whiteSpace*

[*https://www.w3schools.com/xml/schema\_dtypes\_date.asp*](https://www.w3schools.com/xml/schema_dtypes_date.asp)

*XSD Date and Time Data Types*

*Date and time data types are used for values that contain date and time.*

*Date Data Type*

*The date data type is used to specify a date.*

*The date is specified in the following form "YYYY-MM-DD" where:*

* *YYYY indicates the year*
* *MM indicates the month*
* *DD indicates the day*

***Note:****All components are required!*

*The following is an example of a date declaration in a schema:*

*<xs:element name="start" type="xs:date"/>*

*An element in your document might look like this:*

*<start>2002-09-24</start>*

*Time Zones*

*To specify a time zone, you can either enter a date in UTC time by adding a "Z" behind the date - like this:*

*<start>2002-09-24Z</start>*

*or you can specify an offset from the UTC time by adding a positive or negative time behind the date - like this:*

*<start>2002-09-24-06:00</start>  
  
or  
  
<start>2002-09-24+06:00</start>*

*Time Data Type*

*The time data type is used to specify a time.*

*The time is specified in the following form "hh:mm:ss" where:*

* *hh indicates the hour*
* *mm indicates the minute*
* *ss indicates the second*

***Note:****All components are required!*

*The following is an example of a time declaration in a schema:*

*<xs:element name="start" type="xs:time"/>*

*An element in your document might look like this:*

*<start>09:00:00</start>*

*Or it might look like this:*

*<start>09:30:10.5</start>*

*Time Zones*

*To specify a time zone, you can either enter a time in UTC time by adding a "Z" behind the time - like this:*

*<start>09:30:10Z</start>*

*or you can specify an offset from the UTC time by adding a positive or negative time behind the time - like this:*

*<start>09:30:10-06:00</start>  
  
or  
  
<start>09:30:10+06:00</start>*

*DateTime Data Type*

*The dateTime data type is used to specify a date and a time.*

*The dateTime is specified in the following form "YYYY-MM-DDThh:mm:ss" where:*

* *YYYY indicates the year*
* *MM indicates the month*
* *DD indicates the day*
* *T indicates the start of the required time section*
* *hh indicates the hour*
* *mm indicates the minute*
* *ss indicates the second*

***Note:****All components are required!*

*The following is an example of a dateTime declaration in a schema:*

*<xs:element name="startdate" type="xs:dateTime"/>*

*An element in your document might look like this:*

*<startdate>2002-05-30T09:00:00</startdate>*

*Or it might look like this:*

*<startdate>2002-05-30T09:30:10.5</startdate>*

*Time Zones*

*To specify a time zone, you can either enter a dateTime in UTC time by adding a "Z" behind the time - like this:*

*<startdate>2002-05-30T09:30:10Z</startdate>*

*or you can specify an offset from the UTC time by adding a positive or negative time behind the time - like this:*

*<startdate>2002-05-30T09:30:10-06:00</startdate>  
  
or  
  
<startdate>2002-05-30T09:30:10+06:00</startdate>*

*Duration Data Type*

*The duration data type is used to specify a time interval.*

*The time interval is specified in the following form "PnYnMnDTnHnMnS" where:*

* *P indicates the period (required)*
* *nY indicates the number of years*
* *nM indicates the number of months*
* *nD indicates the number of days*
* *T indicates the start of a time section (required if you are going to specify hours, minutes, or seconds)*
* *nH indicates the number of hours*
* *nM indicates the number of minutes*
* *nS indicates the number of seconds*

*The following is an example of a duration declaration in a schema:*

*<xs:element name="period" type="xs:duration"/>*

*An element in your document might look like this:*

*<period>P5Y</period>*

*The example above indicates a period of five years.*

*Or it might look like this:*

*<period>P5Y2M10D</period>*

*The example above indicates a period of five years, two months, and 10 days.*

*Or it might look like this:*

*<period>P5Y2M10DT15H</period>*

*The example above indicates a period of five years, two months, 10 days, and 15 hours.*

*Or it might look like this:*

*<period>PT15H</period>*

*The example above indicates a period of 15 hours.*

*Negative Duration*

*To specify a negative duration, enter a minus sign before the P:*

*<period>-P10D</period>*

*The example above indicates a period of minus 10 days.*

*Date and Time Data Types*

|  |  |
| --- | --- |
| ***Name*** | ***Description*** |
| *date* | *Defines a date value* |
| *dateTime* | *Defines a date and time value* |
| *duration* | *Defines a time interval* |
| *gDay* | *Defines a part of a date - the day (DD)* |
| *gMonth* | *Defines a part of a date - the month (MM)* |
| *gMonthDay* | *Defines a part of a date - the month and day (MM-DD)* |
| *gYear* | *Defines a part of a date - the year (YYYY)* |
| *gYearMonth* | *Defines a part of a date - the year and month (YYYY-MM)* |
| *time* | *Defines a time value* |

*Restrictions on Date Data Types*

*Restrictions that can be used with Date data types:*

* *enumeration*
* *maxExclusive*
* *maxInclusive*
* *minExclusive*
* *minInclusive*
* *pattern*
* *whiteSpace*

[*https://www.w3schools.com/xml/schema\_dtypes\_numeric.asp*](https://www.w3schools.com/xml/schema_dtypes_numeric.asp)

*XSD Numeric Data Types*

*Decimal data types are used for numeric values.*

*Decimal Data Type*

*The decimal data type is used to specify a numeric value.*

*The following is an example of a decimal declaration in a schema:*

*<xs:element name="prize" type="xs:decimal"/>*

*An element in your document might look like this:*

*<prize>999.50</prize>*

*Or it might look like this:*

*<prize>+999.5450</prize>*

*Or it might look like this:*

*<prize>-999.5230</prize>*

*Or it might look like this:*

*<prize>0</prize>*

*Or it might look like this:*

*<prize>14</prize>*

***Note:****The maximum number of decimal digits you can specify is 18.*

*Integer Data Type*

*The integer data type is used to specify a numeric value without a fractional component.*

*The following is an example of an integer declaration in a schema:*

*<xs:element name="prize" type="xs:integer"/>*

*An element in your document might look like this:*

*<prize>999</prize>*

*Or it might look like this:*

*<prize>+999</prize>*

*Or it might look like this:*

*<prize>-999</prize>*

*Or it might look like this:*

*<prize>0</prize>*

*Numeric Data Types*

*Note that all of the data types below derive from the Decimal data type (except for decimal itself)!*

|  |  |
| --- | --- |
| ***Name*** | ***Description*** |
| *byte* | *A signed 8-bit integer* |
| *decimal* | *A decimal value* |
| *int* | *A signed 32-bit integer* |
| *integer* | *An integer value* |
| *long* | *A signed 64-bit integer* |
| *negativeInteger* | *An integer containing only negative values (..,-2,-1)* |
| *nonNegativeInteger* | *An integer containing only non-negative values (0,1,2,..)* |
| *nonPositiveInteger* | *An integer containing only non-positive values (..,-2,-1,0)* |
| *positiveInteger* | *An integer containing only positive values (1,2,..)* |
| *short* | *A signed 16-bit integer* |
| *unsignedLong* | *An unsigned 64-bit integer* |
| *unsignedInt* | *An unsigned 32-bit integer* |
| *unsignedShort* | *An unsigned 16-bit integer* |
| *unsignedByte* | *An unsigned 8-bit integer* |

*Restrictions on Numeric Data Types*

*Restrictions that can be used with Numeric data types:*

* *enumeration*
* *fractionDigits*
* *maxExclusive*
* *maxInclusive*
* *minExclusive*
* *minInclusive*
* *pattern*
* *totalDigits*
* *whiteSpace*

[*https://www.w3schools.com/xml/schema\_dtypes\_misc.asp*](https://www.w3schools.com/xml/schema_dtypes_misc.asp)

*XSD Miscellaneous Data Types*

*Other miscellaneous data types are boolean, base64Binary, hexBinary, float, double, anyURI, QName, and NOTATION.*

*Boolean Data Type*

*The boolean data type is used to specify a true or false value.*

*The following is an example of a boolean declaration in a schema:*

*<xs:attribute name="disabled" type="xs:boolean"/>*

*An element in your document might look like this:*

*<prize disabled="true">999</prize>*

***Note:****Legal values for boolean are true, false, 1 (which indicates true), and 0 (which indicates false).*

*Binary Data Types*

*Binary data types are used to express binary-formatted data.*

*We have two binary data types:*

* *base64Binary (Base64-encoded binary data)*
* *hexBinary (hexadecimal-encoded binary data)*

*The following is an example of a hexBinary declaration in a schema:*

*<xs:element name="blobsrc" type="xs:hexBinary"/>*

*AnyURI Data Type*

*The anyURI data type is used to specify a URI.*

*The following is an example of an anyURI declaration in a schema:*

*<xs:attribute name="src" type="xs:anyURI"/>*

*An element in your document might look like this:*

*<pic src="https://www.w3schools.com/images/smiley.gif" />*

***Note:****If a URI has spaces, replace them with %20.*

*Miscellaneous Data Types*

|  |  |
| --- | --- |
| ***Name*** | ***Description*** |
| *anyURI* |  |
| *base64Binary* |  |
| *boolean* |  |
| *double* |  |
| *float* |  |
| *hexBinary* |  |
| *NOTATION* |  |
| *QName* |  |

*Restrictions on Miscellaneous Data Types*

*Restrictions that can be used with the other data types:*

* *enumeration (a Boolean data type cannot use this constraint)*
* *length (a Boolean data type cannot use this constraint)*
* *maxLength (a Boolean data type cannot use this constraint)*
* *minLength (a Boolean data type cannot use this constraint)*
* *pattern*
* *whiteSpace*

[*https://www.w3schools.com/xml/schema\_elements\_ref.asp*](https://www.w3schools.com/xml/schema_elements_ref.asp)

*XML Schema Reference*

*XSD Elements*

|  |  |
| --- | --- |
| ***Element*** | ***Explanation*** |
| [*all*](https://www.w3schools.com/xml/el_all.asp) | *Specifies that the child elements can appear in any order. Each child element can occur 0 or 1 time* |
| [*annotation*](https://www.w3schools.com/xml/el_annotation.asp) | *Specifies the top-level element for schema comments* |
| [*any*](https://www.w3schools.com/xml/el_any.asp) | *Enables the author to extend the XML document with elements not specified by the schema* |
| [*anyAttribute*](https://www.w3schools.com/xml/el_anyattribute.asp) | *Enables the author to extend the XML document with attributes not specified by the schema* |
| [*appinfo*](https://www.w3schools.com/xml/el_appinfo.asp) | *Specifies information to be used by the application (must go inside annotation)* |
| [*attribute*](https://www.w3schools.com/xml/el_attribute.asp) | *Defines an attribute* |
| [*attributeGroup*](https://www.w3schools.com/xml/el_attributegroup.asp) | *Defines an attribute group to be used in complex type definitions* |
| [*choice*](https://www.w3schools.com/xml/el_choice.asp) | *Allows only one of the elements contained in the <choice> declaration to be present within the containing element* |
| [*complexContent*](https://www.w3schools.com/xml/el_complexcontent.asp) | *Defines extensions or restrictions on a complex type that contains mixed content or elements only* |
| [*complexType*](https://www.w3schools.com/xml/el_complextype.asp) | *Defines a complex type element* |
| [*documentation*](https://www.w3schools.com/xml/el_documentation.asp) | *Defines text comments in a schema (must go inside annotation)* |
| [*element*](https://www.w3schools.com/xml/el_element.asp) | *Defines an element* |
| [*extension*](https://www.w3schools.com/xml/el_extension.asp) | *Extends an existing simpleType or complexType element // гэта рэалізуе механізм спадчыннасці між тыпамі? //* |
| [*field*](https://www.w3schools.com/xml/el_field.asp) | *Specifies an XPath expression that specifies the value used to define an identity constraint* |
| [*group*](https://www.w3schools.com/xml/el_group.asp) | *Defines a group of elements to be used in complex type definitions* |
| [*import*](https://www.w3schools.com/xml/el_import.asp) | *Adds multiple schemas with different target namespace to a document* |
| [*include*](https://www.w3schools.com/xml/el_include.asp) | *Adds multiple schemas with the same target namespace to a document* |
| [*key*](https://www.w3schools.com/xml/el_key.asp) | *Specifies an attribute or element value as a key (unique, non-nullable, and always present) within the containing element in an instance document* |
| [*keyref*](https://www.w3schools.com/xml/el_keyref.asp) | *Specifies that an attribute or element value correspond to those of the specified key or unique element* |
| [*list*](https://www.w3schools.com/xml/el_list.asp) | *Defines a simple type element as a list of values* |
| [*notation*](https://www.w3schools.com/xml/el_notation.asp) | *Describes the format of non-XML data within an XML document* |
| [*redefine*](https://www.w3schools.com/xml/el_redefine.asp) | *Redefines simple and complex types, groups, and attribute groups from an external schema* |
| [*restriction*](https://www.w3schools.com/xml/el_restriction.asp) | *Defines restrictions on a simpleType, simpleContent, or a complexContent* |
| [*schema*](https://www.w3schools.com/xml/el_schema.asp) | *Defines the root element of a schema* |
| [*selector*](https://www.w3schools.com/xml/el_selector.asp) | *Specifies an XPath expression that selects a set of elements for an identity constraint* |
| [*sequence*](https://www.w3schools.com/xml/el_sequence.asp) | *Specifies that the child elements must appear in a sequence. Each child element can occur from 0 to any number of times* |
| [*simpleContent*](https://www.w3schools.com/xml/el_simpleContent.asp) | *Contains extensions or restrictions on a text-only complex type or on a simple type as content and contains no elements* |
| [*simpleType*](https://www.w3schools.com/xml/el_simpletype.asp) | *Defines a simple type and specifies the constraints and information about the values of attributes or text-only elements* |
| [*union*](https://www.w3schools.com/xml/el_union.asp) | *Defines a simple type as a collection (union) of values from specified simple data types* |
| [*unique*](https://www.w3schools.com/xml/el_unique.asp) | *Defines that an element or an attribute value must be unique within the scope* |

*XSD Restrictions/Facets for Datatypes*

[*Look at XSD Restrictions!*](https://www.w3schools.com/xml/schema_facets.asp)

|  |  |
| --- | --- |
| ***Constraint*** | ***Description*** |
| *enumeration* | *Defines a list of acceptable values* |
| *fractionDigits* | *Specifies the maximum number of decimal places allowed. Must be equal to or greater than zero* |
| *length* | *Specifies the exact number of characters or list items allowed. Must be equal to or greater than zero* |
| *maxExclusive* | *Specifies the upper bounds for numeric values (the value must be less than this value)* |
| *maxInclusive* | *Specifies the upper bounds for numeric values (the value must be less than or equal to this value)* |
| *maxLength* | *Specifies the maximum number of characters or list items allowed. Must be equal to or greater than zero* |
| *minExclusive* | *Specifies the lower bounds for numeric values (the value must be greater than this value)* |
| *minInclusive* | *Specifies the lower bounds for numeric values (the value must be greater than or equal to this value)* |
| *minLength* | *Specifies the minimum number of characters or list items allowed. Must be equal to or greater than zero* |
| *pattern* | *Defines the exact sequence of characters that are acceptable* |
| *totalDigits* | *Specifies the maximum number of digits allowed. Must be greater than zero* |
| *whiteSpace* | *Specifies how white space (line feeds, tabs, spaces, and carriage returns) is handled* |

XML Web Services

[*https://www.w3schools.com/xml/xml\_services.asp*](https://www.w3schools.com/xml/xml_services.asp)

*// пачытаем пра веб-сервісы //*

*XML Web Services*

*Web services are web application components.*

*Web services can be published, found, and used on the Web.*

*This tutorial introduces WSDL, SOAP, RDF, and RSS.*

*WSDL*

* *WSDL stands for Web Services Description Language*
* *WSDL is an XML-based language for describing Web services.*
* *WSDL is a W3C recommendation*

*SOAP*

* *SOAP stands for Simple Object Access Protocol*
* *SOAP is an XML based protocol for accessing Web Services.*
* *SOAP is based on XML*
* *SOAP is a W3C recommendation*

*RDF*

* *RDF stands for Resource Description Framework*
* *RDF is a framework for describing resources on the web*
* *RDF is written in XML*
* *RDF is a W3C Recommendation*

*RSS*

* *RSS stands for Really Simple Syndication*
* *RSS allows you to syndicate your site content*
* *RSS defines an easy way to share and view headlines and content*
* *RSS files can be automatically updated*
* *RSS allows personalized views for different sites*
* *RSS is written in XML*

*What You Should Already Know*

*Before you study web services you should have a basic understanding of XML and XML Namespaces.*

*If you want to study these subjects first, please read our*[*XML Tutorial*](https://www.w3schools.com/xml/default.asp)*.*

*Web Services*

* *Web services are application components*
* *Web services communicate using open protocols*
* *Web services are self-contained and self-describing*
* *Web services can be discovered using UDDI*
* *Web services can be used by other applications*
* *HTTP and XML is the basis for Web services*

*Interoperability has Highest Priority*

*When all major platforms could access the Web using Web browsers, different platforms couldn't interact. For these platforms to work together, Web-applications were developed.*

*// асноўная матывацыя і прызначэнне веб-прылажэнняў. Блізка да канцэпцыі Джава //*

*Web-applications are simply applications that run on the web. These are built around the Web browser standards and can be used by any browser on any platform.*

*// рэалізацыя веб-прылажэнняў //*

*Web Services take Web-applications to the Next Level*

*By using Web services, your application can publish its function or message to the rest of the world.*

*Web services use XML to code and to decode data, and SOAP to transport it (using open protocols).*

*With Web services, your accounting department's Win 2k server's billing system can connect with your IT supplier's UNIX server.*

*// асноўнае прызначэнне і перавага веб-сервісаў. Але цікавыя былі бы прыклады. Асабліва цудоўна было бы параўнаць тое, што было да веб-сервісаў, і што стала магчымым з веб-сервісамі //*

*Web Services have Two Types of Uses*

***Reusable application-components.***

*There are things applications need very often. So why make these over and over again?*

*Web services can offer application-components like: currency conversion, weather reports, or even language translation as services.*

*// асноўная ідэя веб-сервісаў. Гэта цудоўная сіла аб’яднання ведаў. //*

***Connect existing software.***

*Web services can help to solve the interoperability problem by giving different applications a way to link their data.*

*With Web services you can exchange data between different applications and different platforms.*

*// вельмі-вельмі палезна. Праўда, здаецца, важна заўжды мець у розуме праблему бяспецы //*

*Any application can have a Web Service component.*

*Web Services can be created regardless of programming language.*

*A Web Service Example*

*In the following example we will use ASP.NET to create a simple Web Service that converts the temperature from Fahrenheit to Celsius, and vice versa:*

*<%@ WebService Language="VBScript" Class="TempConvert" %>  
  
Imports System  
Imports System.Web.Services  
  
Public Class TempConvert :Inherits WebService  
  
<WebMethod()> Public Function FahrenheitToCelsius  
(ByVal Fahrenheit As String) As String  
  dim fahr  
  fahr=trim(replace(Fahrenheit,",","."))  
  if fahr="" or IsNumeric(fahr)=false then return "Error"  
  return ((((fahr) - 32) / 9) \* 5)  
end function  
  
<WebMethod()> Public Function CelsiusToFahrenheit  
(ByVal Celsius As String) As String  
  dim cel  
  cel=trim(replace(Celsius,",","."))  
  if cel="" or IsNumeric(cel)=false then return "Error"  
  return ((((cel) \* 9) / 5) + 32)  
end function  
  
end class*

*This document is saved as an .asmx file. This is the ASP.NET file extension for XML Web Services.*

*Example Explained*

***Note:****To run this example, you will need a .NET server.*

*The first line in the example states that this is a Web Service, written in VBScript, and has the class name "TempConvert":*

*<%@ WebService Language="VBScript" Class="TempConvert" %>*

*The next lines import the namespace "System.Web.Services" from the .NET framework:*

*Imports System  
Imports System.Web.Services*

*The next line defines that the "TempConvert" class is a WebService class type:*

*Public Class TempConvert :Inherits WebService*

*The next steps are basic VB programming. This application has two functions. One to convert from Fahrenheit to Celsius, and one to convert from Celsius to Fahrenheit.*

*The only difference from a normal application is that this function is defined as a "WebMethod()".*

*Use "WebMethod()" to convert the functions in your application into web services:*

*<WebMethod()> Public Function FahrenheitToCelsius  
(ByVal Fahrenheit As String) As String  
  dim fahr  
  fahr=trim(replace(Fahrenheit,",","."))  
  if fahr="" or IsNumeric(fahr)=false then return "Error"  
  return ((((fahr) - 32) / 9) \* 5)  
end function  
  
<WebMethod()> Public Function CelsiusToFahrenheit  
(ByVal Celsius As String) As String  
  dim cel  
  cel=trim(replace(Celsius,",","."))  
  if cel="" or IsNumeric(cel)=false then return "Error"  
  return ((((cel) \* 9) / 5) + 32)  
end function*

*Then, end the class:*

*end class*

*Publish the .asmx file on a server with .NET support, and you will have your first working Web Service.*

*Put the Web Service on Your Web Site*

*Using a form and the HTTP POST method, you can put the web service on your site, like this:*

*…*

*// схема мне зразумела. Але код віжуал бэйсіка, відаць, тварэння Купера, мне не вельмі зразумелы //*

*Конец формы*

*How To Do It*

*Here is the code to add the Web Service to a web page:*

*<form action='tempconvert.asmx/FahrenheitToCelsius'  
method="post" target="\_blank">  
<table>  
  <tr>  
    <td>Fahrenheit to Celsius:</td>  
    <td>  
    <input class="frmInput" type="text" size="30" name="Fahrenheit">  
    </td>  
  </tr>  
  <tr>  
    <td></td>  
    <td align="right">  
     <input type="submit" value="Submit" class="button">  
     </td>  
  </tr>  
</table>  
</form>  
  
<form action='tempconvert.asmx/CelsiusToFahrenheit'  
method="post" target="\_blank">  
<table>  
  <tr>  
    <td>Celsius to Fahrenheit:</td>  
    <td>  
    <input class="frmInput" type="text" size="30" name="Celsius">  
    </td>  
  </tr>  
  <tr>  
    <td></td>  
    <td align="right">  
    <input type="submit" value="Submit" class="button">  
    </td>  
  </tr>  
</table>  
</form>*

*// вылучаныя мною радкі, відаць, злучаюць функцыі веб-сервіса, апісаныя ў ASP.NET, з формамі. Пачытаць пра дадзеныя элементы html //*

*Substitute the "tempconvert.asmx" with the address of your web service like:*

[*http://www.example.com/xml/tempconvert.asmx*](http://www.example.com/xml/tempconvert.asmx)

*// а як іншым людзям карыстацца веб-сервісамі? //*

*// іншыя людзі могуць купіць гэты веб-сервіс і з лёгкасцю ўнядрыць яго ў свае веб-старонкі? //*

*// а іншыя прылажэнні могуць выкарыстоўваць апісанне веб-сервіса, каб інтэрпрэтаваць яго дадзены? //*

[*https://www.w3schools.com/xml/xml\_wsdl.asp*](https://www.w3schools.com/xml/xml_wsdl.asp)

*XML WSDL*

* *WSDL stands for Web Services Description Language*
* *WSDL is used to describe web services*
* *WSDL is written in XML*
* *WSDL is a W3C recommendation from 26. June 2007*

*WSDL Documents*

*An WSDL document describes a web service. It specifies the location of the service, and the methods of the service, using these major elements:*

|  |  |
| --- | --- |
| ***Element*** | ***Description*** |
| *<types>* | *Defines the (XML Schema) data types used by the web service* |
| *<message>* | *Defines the data elements for each operation* |
| *<portType>* | *Describes the operations that can be performed and the messages involved.* |
| *<binding>* | *Defines the protocol and data format for each port type* |

*// апошні радок, відаць, шмат што кажа пра сецевыя тэхналогіі //*

*The main structure of a WSDL document looks like this:*

*<definitions>  
  
<types>  
  data type definitions........  
</types>  
  
<message>  
  definition of the data being communicated....  
</message>  
  
<portType>  
  set of operations......  
</portType>  
  
<binding>  
  protocol and data format specification....  
</binding>  
  
</definitions>*

*WSDL Example*

*This is a simplified fraction of a WSDL document:*

*<message name="getTermRequest">  
  <part name="term" type="xs:string"/>  
</message>  
  
<message name="getTermResponse">  
  <part name="value" type="xs:string"/>  
</message>  
  
<portType name="glossaryTerms">  
  <operation name="getTerm">  
    <input message="getTermRequest"/>  
    <output message="getTermResponse"/>  
  </operation>  
</portType>*

*In this example the****<portType>****element defines "glossaryTerms" as the name of a****port****, and "getTerm" as the name of an****operation****.*

*The "getTerm" operation has an****input message****called "getTermRequest" and an****output message****called "getTermResponse".*

*The****<message>****elements define the****parts****of each message and the associated data types.*

*The <portType> Element*

*The <portType> element defines****a web service****, the****operations****that can be performed, and the****messages****that are involved.*

*The request-response type is the most common operation type, but WSDL defines four types:*

|  |  |
| --- | --- |
| ***Type*** | ***Definition*** |
| *One-way* | *The operation can receive a message but will not return a response* |
| *Request-response* | *The operation can receive a request and will return a response* |
| *Solicit-response* | *The operation can send a request and will wait for a response* |
| *Notification* | *The operation can send a message but will not wait for a response* |

*WSDL One-Way Operation*

*A one-way operation example:*

*<message name="newTermValues">  
  <part name="term" type="xs:string"/>  
  <part name="value" type="xs:string"/>  
</message>  
  
<portType name="glossaryTerms">  
  <operation name="setTerm">  
    <input name="newTerm" message="newTermValues"/>  
  </operation>  
</portType >*

*// спачатку, я так разумею, вызначаны фармат паведамлення. А потым функцыя на серверы, якая выконваецца пры выкліку. Але я не магу злучыць параметры гэтай функцыі і параметры, што дасылаюць з паведамленнем //*

*In the example above, the portType "glossaryTerms" defines a one-way operation called "setTerm".*

*The "setTerm" operation allows input of new glossary terms messages using a "newTermValues" message with the input parameters "term" and "value". However, no output is defined for the operation.*

*// а дзе апісаная рэалізацыя? У самім прылажэнні? Лепей зразумець, у чым перавага WSDL? Адчуць, што было бы без яго, а што ёсць з ім //*

*// what is new term? //*

*WSDL Request-Response Operation*

*A request-response operation example:*

*<message name="getTermRequest">  
  <part name="term" type="xs:string"/>  
</message>  
  
<message name="getTermResponse">  
  <part name="value" type="xs:string"/>  
</message>  
  
<portType name="glossaryTerms">  
  <operation name="getTerm">  
    <input message="getTermRequest"/>  
    <output message="getTermResponse"/>  
  </operation>  
</portType>*

*In the example above, the portType "glossaryTerms" defines a request-response operation called "getTerm".*

*The "getTerm" operation requires an input message called "getTermRequest" with a parameter called "term", and will return an output message called "getTermResponse" with a parameter called "value".*

*// прыклад мне, у цэлым зразумелы, але я пакуль выяўна не адчуваю перавагі дадзенай тэхналогіі. А таксама я не магу рэалізацаваць веб-сервіс. Пакуль не разумею, як //*

*WSDL Binding to SOAP*

*WSDL bindings defines the message format and protocol details for a web service.*

*A request-response operation example:*

*<message name="getTermRequest">  
  <part name="term" type="xs:string"/>  
</message>  
  
<message name="getTermResponse">  
  <part name="value" type="xs:string"/>  
</message>  
  
<portType name="glossaryTerms">  
  <operation name="getTerm">  
    <input message="getTermRequest"/>  
    <output message="getTermResponse"/>  
  </operation>  
</portType>  
  
<binding type="glossaryTerms" name="b1">  
   <soap:binding style="document"  
   transport="http://schemas.xmlsoap.org/soap/http" />  
   <operation>  
     <soap:operation soapAction="http://example.com/getTerm"/>  
     <input><soap:body use="literal"/></input>  
     <output><soap:body use="literal"/></output>  
  </operation>  
</binding>*

*The****binding****element has two attributes - name and type.*

*The name attribute (you can use any name you want) defines the name of the binding, and the type attribute points to the port for the binding, in this case the "glossaryTerms" port.*

*The****soap:binding****element has two attributes - style and transport.*

*The style attribute can be "rpc" or "document". In this case we use document. The transport attribute defines the SOAP protocol to use. In this case we use HTTP.*

*The****operation****element defines each operation that the portType exposes.*

*For each operation the corresponding SOAP action has to be defined. You must also specify how the input and output are encoded. In this case we use "literal".*

*// тэханлогію разумею толькі прыблізна… паразважаць… даследваць… павучыцца… //*

[*https://www.w3schools.com/xml/xml\_soap.asp*](https://www.w3schools.com/xml/xml_soap.asp)

*XML Soap*

* *SOAP stands for****S****imple****O****bject****A****ccess****P****rotocol*
* *SOAP is an application communication protocol*
* *SOAP is a format for sending and receiving messages*
* *SOAP is platform independent*
* *SOAP is based on XML*
* *SOAP is a W3C recommendation*

*Why SOAP?*

*It is important for web applications to be able to communicate over the Internet.*

*The best way to communicate between applications is over HTTP, because HTTP is supported by all Internet browsers and servers. SOAP was created to accomplish this.*

*SOAP provides a way to communicate between applications running on different operating systems, with different technologies and programming languages.*

*SOAP Building Blocks*

*A SOAP message is an ordinary XML document containing the following elements:*

* *An Envelope element that identifies the XML document as a SOAP message*
* *A Header element that contains header information*
* *A Body element that contains call and response information*
* *A Fault element containing errors and status information*

*All the elements above are declared in the default namespace for the SOAP envelope:*

[*http://www.w3.org/2003/05/soap-envelope/*](http://www.w3.org/2003/05/soap-envelope/)

*and the default namespace for SOAP encoding and data types is:*

[*http://www.w3.org/2003/05/soap-encoding*](http://www.w3.org/2003/05/soap-encoding)

*Syntax Rules*

*Here are some important syntax rules:*

* *A SOAP message MUST be encoded using XML*
* *A SOAP message MUST use the SOAP Envelope namespace*
* *A SOAP message MUST use the SOAP Encoding namespace*
* *A SOAP message must NOT contain a DTD reference*
* *A SOAP message must NOT contain XML Processing Instructions*

*Skeleton SOAP Message*

*<?xml version="1.0"?>  
  
<soap:Envelope  
xmlns:soap="http://www.w3.org/2003/05/soap-envelope/"  
soap:encodingStyle="http://www.w3.org/2003/05/soap-encoding">  
  
<soap:Header>  
...  
</soap:Header>  
  
<soap:Body>  
...  
  <soap:Fault>  
  ...  
  </soap:Fault>  
</soap:Body>  
  
</soap:Envelope>*

*The SOAP Envelope Element*

*The required SOAP Envelope element is the root element of a SOAP message. This element defines the XML document as a SOAP message.*

*Example*

*<?xml version="1.0"?>  
  
<soap:Envelope  
xmlns:soap="http://www.w3.org/2003/05/soap-envelope/"  
soap:encodingStyle="http://www.w3.org/2003/05/soap-encoding">  
  ...  
  Message information goes here  
  ...  
</soap:Envelope>*

*The xmlns:soap Namespace*

*Notice the xmlns:soap namespace in the example above. It should always have the value of: "http://www.w3.org/2003/05/soap-envelope/".*

*The namespace defines the Envelope as a SOAP Envelope.*

*If a different namespace is used, the application generates an error and discards the message.*

*The encodingStyle Attribute*

*The encodingStyle attribute is used to define the data types used in the document. This attribute may appear on any SOAP element, and applies to the element's contents and all child elements.*

*A SOAP message has no default encoding.*

*Syntax*

*soap:encodingStyle="URI"*

*Example*

*<?xml version="1.0"?>  
  
<soap:Envelope  
xmlns:soap="http://www.w3.org/2003/05/soap-envelope/"  
soap:encodingStyle="http://www.w3.org/2003/05/soap-encoding">  
  ...  
  Message information goes here  
  ...  
</soap:Envelope>*

*The SOAP Header Element*

*The optional SOAP Header element contains application-specific information (like authentication, payment, etc) about the SOAP message.*

*If the Header element is present, it must be the first child element of the Envelope element.*

***Note:****All immediate child elements of the Header element must be namespace-qualified.*

*<?xml version="1.0"?>  
  
<soap:Envelope  
xmlns:soap="http://www.w3.org/2003/05/soap-envelope/"  
soap:encodingStyle="http://www.w3.org/2003/05/soap-encoding">  
  
<soap:Header>  
  <m:Trans xmlns:m="https://www.w3schools.com/transaction/"  
  soap:mustUnderstand="1">234  
  </m:Trans>  
</soap:Header>  
...  
...  
</soap:Envelope>*

*The example above contains a header with a "Trans" element, a "mustUnderstand" attribute with a value of 1, and a value of 234.*

*SOAP defines three attributes in the default namespace. These attributes are: mustUnderstand, actor, and encodingStyle.*

*The attributes defined in the SOAP Header defines how a recipient should process the SOAP message.*

*The mustUnderstand Attribute*

*The SOAP mustUnderstand attribute can be used to indicate whether a header entry is mandatory or optional for the recipient to process.*

*If you add mustUnderstand="1" to a child element of the Header element it indicates that the receiver processing the Header must recognize the element. If the receiver does not recognize the element it will fail when processing the Header.*

*Syntax*

*soap:mustUnderstand="0|1"*

*Example*

*<?xml version="1.0"?>  
  
<soap:Envelope  
xmlns:soap="http://www.w3.org/2003/05/soap-envelope/"  
soap:encodingStyle="http://www.w3.org/2003/05/soap-encoding">  
  
<soap:Header>  
  <m:Trans xmlns:m="https://www.w3schools.com/transaction/"  
  soap:mustUnderstand="1">234  
  </m:Trans>  
</soap:Header>  
...  
...  
</soap:Envelope>*

*// гэта тычыцца любога атрыбута ў хедэры //*

*The actor Attribute*

*A SOAP message may travel from a sender to a receiver by passing different endpoints along the message path. However, not all parts of a SOAP message may be intended for the ultimate endpoint, instead, it may be intended for one or more of the endpoints on the message path.*

*// message path – important concept of SOAP //*

*The SOAP actor attribute is used to address the Header element to a specific endpoint.*

*Syntax*

*soap:actor="URI"*

*Example*

*<?xml version="1.0"?>  
  
<soap:Envelope  
xmlns:soap="http://www.w3.org/2003/05/soap-envelope/"  
soap:encodingStyle="http://www.w3.org/2003/05/soap-encoding">  
  
<soap:Header>  
  <m:Trans xmlns:m="https://www.w3schools.com/transaction/"  
  soap:actor="https://www.w3schools.com/appml/">234  
  </m:Trans>  
</soap:Header>  
...  
...  
</soap:Envelope>*

*// it is also used for every element in header //*

*The encodingStyle Attribute*

*The encodingStyle attribute is used to define the data types used in the document. This attribute may appear on any SOAP element, and it will apply to that element's contents and all child elements.*

*A SOAP message has no default encoding.*

*Syntax*

*soap:encodingStyle="URI"*

*// я не да канца зразумеў, што такое стыль дакумента //*

*The SOAP Body Element*

*The required SOAP Body element contains the actual SOAP message intended for the ultimate endpoint of the message.*

*Immediate child elements of the SOAP Body element may be namespace-qualified.*

*Example*

*<?xml version="1.0"?>  
  
<soap:Envelope  
xmlns:soap="http://www.w3.org/2003/05/soap-envelope/"  
soap:encodingStyle="http://www.w3.org/2003/05/soap-encoding">  
  
<soap:Body>  
  <m:GetPrice xmlns:m="https://www.w3schools.com/prices">  
    <m:Item>Apples</m:Item>  
  </m:GetPrice>  
</soap:Body>  
  
</soap:Envelope>*

*The example above requests the price of apples. Note that the m:GetPrice and the Item elements above are application-specific elements. They are not a part of the SOAP namespace.*

*// прылажэнне інтэрпрэтуе інфармацыю з паведамлення, апрацоўваючы спецыфічныя для яго тэгі і выконвае функцыі. Я зразумеў, як працуе SOAP! //*

*A SOAP response could look something like this:*

*<?xml version="1.0"?>  
  
<soap:Envelope  
xmlns:soap="http://www.w3.org/2003/05/soap-envelope/"  
soap:encodingStyle="http://www.w3.org/2003/05/soap-encoding">  
  
<soap:Body>  
  <m:GetPriceResponse xmlns:m="https://www.w3schools.com/prices">  
    <m:Price>1.90</m:Price>  
  </m:GetPriceResponse>  
</soap:Body>  
  
</soap:Envelope>*

*The SOAP Fault Element*

*The optional SOAP Fault element is used to indicate error messages.*

*The SOAP Fault element holds errors and status information for a SOAP message.*

*If a Fault element is present, it must appear as a child element of the Body element. A Fault element can only appear once in a SOAP message.*

*The SOAP Fault element has the following sub elements:*

|  |  |
| --- | --- |
| ***Sub Element*** | ***Description*** |
| *<faultcode>* | *A code for identifying the fault* |
| *<faultstring>* | *A human readable explanation of the fault* |
| *<faultactor>* | *Information about who caused the fault to happen* |
| *<detail>* | *Holds application specific error information related to the Body element* |

*SOAP Fault Codes*

*The faultcode values defined below must be used in the faultcode element when describing faults:*

|  |  |
| --- | --- |
| ***Error*** | ***Description*** |
| *VersionMismatch* | *Found an invalid namespace for the SOAP Envelope element* |
| *MustUnderstand* | *An immediate child element of the Header element, with the mustUnderstand attribute set to "1", was not understood* |
| *Client* | *The message was incorrectly formed or contained incorrect information* |
| *Server* | *There was a problem with the server so the message could not proceed* |

*The HTTP Protocol*

*HTTP communicates over TCP/IP. An HTTP client connects to an HTTP server using TCP. After establishing a connection, the client can send an HTTP request message to the server:*

*POST /item HTTP/1.1  
Host: 189.123.255.239  
Content-Type: text/plain  
Content-Length: 200*

*The server then processes the request and sends an HTTP response back to the client. The response contains a status code that indicates the status of the request:*

*200 OK  
Content-Type: text/plain  
Content-Length: 200*

*In the example above, the server returned a status code of 200. This is the standard success code for HTTP.*

*If the server could not decode the request, it could have returned something like this:*

*400 Bad Request  
Content-Length: 0*

*SOAP Binding*

*The SOAP specification defines the structure of the SOAP messages, not how they are exchanged. This gap is filled by what is called "SOAP Bindings". SOAP bindings are mechanisms which allow SOAP messages to be effectively exchanged using a transport protocol.*

*Most SOAP implementations provide bindings for common transport protocols, such as HTTP or SMTP.*

*HTTP is synchronous and widely used. A SOAP HTTP request specifies at least two HTTP headers: Content-Type and Content-Length.*

*SMTP is asynchronous and is used in last resort or particular cases.*

*Java implementations of SOAP usually provide a specific binding for the JMS (Java Messaging System) protocol.*

*Content-Type*

*The Content-Type header for a SOAP request and response defines the MIME type for the message and the character encoding (optional) used for the XML body of the request or response.*

*Syntax*

*Content-Type: MIMEType; charset=character-encoding*

*Example*

*POST /item HTTP/1.1  
Content-Type: application/soap+xml; charset=utf-8*

*Content-Length*

*The Content-Length header for a SOAP request and response specifies the number of bytes in the body of the request or response.*

*Syntax*

*Content-Length: bytes*

*Example*

*POST /item HTTP/1.1  
Content-Type: application/soap+xml; charset=utf-8  
Content-Length: 250*

*A SOAP Example*

*In the example below, a GetStockPrice request is sent to a server. The request has a StockName parameter, and a Price parameter that will be returned in the response. The namespace for the function is defined in "http://www.example.org/stock".*

*A SOAP request:*

*POST /InStock HTTP/1.1  
Host: www.example.org  
Content-Type: application/soap+xml; charset=utf-8  
Content-Length: nnn  
  
<?xml version="1.0"?>  
  
<soap:Envelope  
xmlns:soap="http://www.w3.org/2003/05/soap-envelope/"  
soap:encodingStyle="http://www.w3.org/2003/05/soap-encoding">  
  
<soap:Body xmlns:m="http://www.example.org/stock">  
  <m:GetStockPrice>  
    <m:StockName>IBM</m:StockName>  
  </m:GetStockPrice>  
</soap:Body>  
  
</soap:Envelope>*

*The SOAP response:*

*HTTP/1.1 200 OK  
Content-Type: application/soap+xml; charset=utf-8  
Content-Length: nnn  
  
<?xml version="1.0"?>  
  
<soap:Envelope  
xmlns:soap="http://www.w3.org/2003/05/soap-envelope/"  
soap:encodingStyle="http://www.w3.org/2003/05/soap-encoding">  
  
<soap:Body xmlns:m="http://www.example.org/stock">  
  <m:GetStockPriceResponse>  
    <m:Price>34.5</m:Price>  
  </m:GetStockPriceResponse>  
</soap:Body>  
  
</soap:Envelope>*

*// я зараз пачаў значна лепей разумець сецевыя тэхналогіі //*

[*https://www.w3schools.com/xml/xml\_rdf.asp*](https://www.w3schools.com/xml/xml_rdf.asp)

*XML RDF*

*RDF Document Example*

*<?xml version="1.0"?>  
  
<rdf:RDF  
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"  
xmlns:si="https://www.w3schools.com/rdf/">  
  
<rdf:Description rdf:about="https://www.w3schools.com">  
  <si:title>W3Schools</si:title>  
  <si:author>Jan Egil Refsnes</si:author>  
</rdf:Description>  
  
</rdf:RDF>*

*What is RDF?*

* *RDF stands for****R****esource****D****escription****F****ramework*
* *RDF is a framework for describing resources on the web*
* *RDF is designed to be read and understood by computers*
* *RDF is not designed for being displayed to people*
* *RDF is written in XML*
* *RDF is a part of the W3C's Semantic Web Activity*
* *RDF is a W3C Recommendation from 10. February 2004*

*RDF - Examples of Use*

* *Describing properties for shopping items, such as price and availability*
* *Describing time schedules for web events*
* *Describing information about web pages (content, author, created and modified date)*
* *Describing content and rating for web pictures*
* *Describing content for search engines*
* *Describing electronic libraries*

*RDF is Designed to be Read by Computers*

*RDF was designed to provide a common way to describe information so it can be read and understood by computer applications.*

*RDF descriptions are not designed to be displayed on the web.*

*RDF is Written in XML*

*RDF documents are written in XML. The XML language used by RDF is called RDF/XML.*

*By using XML, RDF information can easily be exchanged between different types of computers using different types of operating systems and application languages.*

*RDF and "The Semantic Web"*

*The RDF language is a part of the W3C's Semantic Web Activity. W3C's "Semantic Web Vision" is a future where:*

* *Web information has exact meaning*
* *Web information can be understood and processed by computers*
* *Computers can integrate information from the web*

*// гэта, сапраўды, вялікая будучыня веб-тэхналогій, якая не за гарамі //*

*// intbu this conception //*

*RDF uses Web identifiers (URIs) to identify resources.*

*RDF describes resources with properties and property values.*

*RDF Resource, Property, and Property Value*

*RDF identifies things using Web identifiers (URIs), and describes resources with properties and property values.*

*Explanation of Resource, Property, and Property value:*

* *A****Resource****is anything that can have a URI, such as "https://www.w3schools.com/rdf"*
* *A****Property****is a Resource that has a name, such as "author" or "homepage"*
* *A****Property value****is the value of a Property, such as "Jan Egil Refsnes" or "https://www.w3schools.com" (note that a property value can be another resource)*

*The following RDF document could describe the resource "https://www.w3schools.com/rdf":*

*<?xml version="1.0"?>  
  
<RDF>  
  <Description about="https://www.w3schools.com/rdf">  
    <author>Jan Egil Refsnes</author>  
    <homepage>https://www.w3schools.com</homepage>  
  </Description>  
</RDF>*

*The example above is simplified. Namespaces are omitted.*

*RDF Statements*

*The combination of a Resource, a Property, and a Property value forms a****Statement****(known as the****subject, predicate and object****of a Statement).*

*Let's look at some example statements to get a better understanding:*

*Statement: "The author of https://www.w3schools.com/rdf is Jan Egil Refsnes".*

* *The subject of the statement above is: https://www.w3schools.com/rdf*
* *The predicate is: author*
* *The object is: Jan Egil Refsnes*

*Statement: "The homepage of https://www.w3schools.com/rdf is https://www.w3schools.com".*

* *The subject of the statement above is: https://www.w3schools.com/rdf*
* *The predicate is: homepage*
* *The object is: https://www.w3schools.com*

*RDF Example*

*Here are two records from a CD-list:*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ***Title*** | ***Artist*** | ***Country*** | ***Company*** | ***Price*** | ***Year*** |
| *Empire Burlesque* | *Bob Dylan* | *USA* | *Columbia* | *10.90* | *1985* |
| *Hide your heart* | *Bonnie Tyler* | *UK* | *CBS Records* | *9.90* | *1988* |

*Below is a few lines from an RDF document:*

*<?xml version="1.0"?>  
  
<rdf:RDF  
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"  
xmlns:cd="http://www.recshop.fake/cd#">  
  
<rdf:Description  
rdf:about="http://www.recshop.fake/cd/Empire Burlesque">  
  <cd:artist>Bob Dylan</cd:artist>  
  <cd:country>USA</cd:country>  
  <cd:company>Columbia</cd:company>  
  <cd:price>10.90</cd:price>  
  <cd:year>1985</cd:year>  
</rdf:Description>  
  
<rdf:Description  
rdf:about="http://www.recshop.fake/cd/Hide your heart">  
  <cd:artist>Bonnie Tyler</cd:artist>  
  <cd:country>UK</cd:country>  
  <cd:company>CBS Records</cd:company>  
  <cd:price>9.90</cd:price>  
  <cd:year>1988</cd:year>  
</rdf:Description>  
.  
.*

*.  
</rdf:RDF>*

*The first line of the RDF document is the XML declaration. The XML declaration is followed by the root element of RDF documents:****<rdf:RDF>****.*

*// мы проста абрамляем xml-дакумент тэгамі rdf //*

*The****xmlns:rdf****namespace, specifies that elements with the rdf prefix are from the namespace "http://www.w3.org/1999/02/22-rdf-syntax-ns#".*

*The****xmlns:cd****namespace, specifies that elements with the cd prefix are from the namespace "http://www.recshop.fake/cd#".*

*The****<rdf:Description>****element contains the description of the resource identified by the****rdf:about****attribute.*

*The elements:****<cd:artist>, <cd:country>, <cd:company>,****etc. are properties of the resource.*

*RDF Online Validator*

[*W3C's RDF Validation Service*](http://www.w3.org/RDF/Validator/)*is useful when learning RDF. Here you can experiment with RDF files.*

*The online RDF Validator parses your RDF document, checks your syntax, and generates tabular and graphical views of your RDF document.*

*Copy and paste the example below into W3C's RDF validator:*

*<?xml version="1.0"?>  
  
<rdf:RDF  
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"  
xmlns:si="https://www.w3schools.com/rdf/">  
<rdf:Description rdf:about="https://www.w3schools.com">  
  <si:title>W3Schools.com</si:title>  
  <si:author>Jan Egil Refsnes</si:author>  
</rdf:Description>  
</rdf:RDF>*

*When you parse the example above,*[*the result will look something like this*](https://www.w3schools.com/xml/RDFvalidationresult.htm)*.*

*RDF Elements*

*The main elements of RDF are the root element, <RDF>, and the <Description> element, which identifies a resource.*

*The <rdf:RDF> Element*

*<rdf:RDF> is the root element of an RDF document. It defines the XML document to be an RDF document. It also contains a reference to the RDF namespace:*

*<?xml version="1.0"?>  
  
<rdf:RDF  
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">  
  ...Description goes here...  
</rdf:RDF>*

*The <rdf:Description> Element*

*The <rdf:Description> element identifies a resource with the about attribute.*

*The <rdf:Description> element contains elements that describe the resource:*

*<?xml version="1.0"?>  
  
<rdf:RDF  
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"  
xmlns:cd="http://www.recshop.fake/cd#">  
  
<rdf:Description  
rdf:about="http://www.recshop.fake/cd/Empire Burlesque">  
  <cd:artist>Bob Dylan</cd:artist>  
  <cd:country>USA</cd:country>  
  <cd:company>Columbia</cd:company>  
  <cd:price>10.90</cd:price>  
  <cd:year>1985</cd:year>  
</rdf:Description>  
  
</rdf:RDF>*

*The elements, artist, country, company, price, and year, are defined in the http://www.recshop.fake/cd# namespace. This namespace is outside RDF (and not a part of RDF). RDF defines only the framework. The elements, artist, country, company, price, and year, must be defined by someone else (company, organization, person, etc).*

*// гэтыя тэхналогіі ёсць яркі прыклад таго, што завецца каркасам //*

*Properties as Attributes*

*The property elements can also be defined as attributes (instead of elements):*

*<?xml version="1.0"?>  
  
<rdf:RDF  
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"  
xmlns:cd="http://www.recshop.fake/cd#">  
  
<rdf:Description  
rdf:about="http://www.recshop.fake/cd/Empire Burlesque"  
cd:artist="Bob Dylan" cd:country="USA"  
cd:company="Columbia" cd:price="10.90"  
cd:year="1985" />  
  
</rdf:RDF>*

*Properties as Resources*

*The property elements can also be defined as resources:*

*<?xml version="1.0"?>  
  
<rdf:RDF  
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"  
xmlns:cd="http://www.recshop.fake/cd#">  
  
<rdf:Description  
rdf:about="http://www.recshop.fake/cd/Empire Burlesque">  
  <cd:artist rdf:resource="http://www.recshop.fake/cd/dylan" />  
  ...  
  ...  
</rdf:Description>  
  
</rdf:RDF>*

*In the example above, the property artist does not have a value, but a reference to a resource containing information about the artist.*

*RDF Containers*

*RDF containers are used to describe group of things.*

*The following RDF elements are used to describe groups: <Bag>, <Seq>, and <Alt>.*

*The <rdf:Bag> Element*

*The <rdf:Bag> element is used to describe a list of values that do not have to be in a specific order.*

*The <rdf:Bag> element may contain duplicate values.*

*Example*

*<?xml version="1.0"?>  
  
<rdf:RDF  
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"  
xmlns:cd="http://www.recshop.fake/cd#">  
  
<rdf:Description  
rdf:about="http://www.recshop.fake/cd/Beatles">  
  <cd:artist>  
    <rdf:Bag>  
      <rdf:li>John</rdf:li>  
      <rdf:li>Paul</rdf:li>  
      <rdf:li>George</rdf:li>  
      <rdf:li>Ringo</rdf:li>  
    </rdf:Bag>  
  </cd:artist>  
</rdf:Description>  
  
</rdf:RDF>*

*The <rdf:Seq> Element*

*The <rdf:Seq> element is used to describe an ordered list of values (For example, in alphabetical order).*

*The <rdf:Seq> element may contain duplicate values.*

*Example*

*<?xml version="1.0"?>  
  
<rdf:RDF  
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"  
xmlns:cd="http://www.recshop.fake/cd#">  
  
<rdf:Description  
rdf:about="http://www.recshop.fake/cd/Beatles">  
  <cd:artist>  
    <rdf:Seq>  
      <rdf:li>George</rdf:li>  
      <rdf:li>John</rdf:li>  
      <rdf:li>Paul</rdf:li>  
      <rdf:li>Ringo</rdf:li>  
    </rdf:Seq>  
  </cd:artist>  
</rdf:Description>  
  
</rdf:RDF>*

*The <rdf:Alt> Element*

*The <rdf:Alt> element is used to describe a list of alternative values (the user can select only one of the values).*

*Example*

*<?xml version="1.0"?>  
  
<rdf:RDF  
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"  
xmlns:cd="http://www.recshop.fake/cd#">  
  
<rdf:Description  
rdf:about="http://www.recshop.fake/cd/Beatles">  
  <cd:format>  
    <rdf:Alt>  
      <rdf:li>CD</rdf:li>  
      <rdf:li>Record</rdf:li>  
      <rdf:li>Tape</rdf:li>  
    </rdf:Alt>  
  </cd:format>  
</rdf:Description>  
  
</rdf:RDF>*

*RDF Terms*

*In the examples above we have talked about "list of values" when describing the container elements. In RDF these "list of values" are called members.*

*So, we have the following:*

* *A container is a resource that contains things*
* *The contained things are called members (not list of values)*

*RDF Collections*

*RDF collections describe groups that can ONLY contain the specified members.*

*The rdf:parseType="Collection" Attribute*

*As seen in the previous chapter, a container says that the containing resources are members - it does not say that other members are not allowed.*

*RDF collections are used to describe groups that can ONLY contain the specified members.*

*A collection is described by the attribute rdf:parseType="Collection".*

*Example*

*<?xml version="1.0"?>  
  
<rdf:RDF  
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"  
xmlns:cd="http://recshop.fake/cd#">  
  
<rdf:Description  
rdf:about="http://recshop.fake/cd/Beatles">  
  <cd:artist rdf:parseType="Collection">  
    <rdf:Description rdf:about="http://recshop.fake/cd/Beatles/George"/>  
    <rdf:Description rdf:about="http://recshop.fake/cd/Beatles/John"/>  
    <rdf:Description rdf:about="http://recshop.fake/cd/Beatles/Paul"/>  
    <rdf:Description rdf:about="http://recshop.fake/cd/Beatles/Ringo"/>  
  </cd:artist>  
</rdf:Description>  
  
</rdf:RDF>*

*RDF Schema and Application Classes*

*RDF Schema (RDFS) is an extension to RDF.*

*RDF describes resources with classes, properties, and values.*

*In addition, RDF also needs a way to define application-specific classes and properties. Application-specific classes and properties must be defined using extensions to RDF.*

*One such extension is RDF Schema.*

*RDF Schema (RDFS)*

*RDF Schema does not provide actual application-specific classes and properties.*

*Instead RDF Schema provides the framework to describe application-specific classes and properties.*

*Classes in RDF Schema are much like classes in object oriented programming languages. This allows resources to be defined as instances of classes, and subclasses of classes.*

*RDFS Example*

*The following example demonstrates some of the RDFS facilities:*

*<?xml version="1.0"?>  
  
<rdf:RDF  
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"  
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"  
xml:base="http://www.animals.fake/animals#">  
  
<rdf:Description rdf:ID="animal">  
  <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>  
</rdf:Description>  
  
<rdf:Description rdf:ID="horse">  
  <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>  
  <rdfs:subClassOf rdf:resource="#animal"/>  
</rdf:Description>  
  
</rdf:RDF>*

*In the example above, the resource "horse" is a subclass of the class "animal".*

*// так гэта тое, што аплікэйшн спесыфік? //*

*Example Abbreviated*

*Since an RDFS class is an RDF resource we can abbreviate the example above by using rdfs:Class instead of rdf:Description, and drop the rdf:type information:*

*<?xml version="1.0"?>  
  
<rdf:RDF  
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"  
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"  
xml:base="http://www.animals.fake/animals#">  
  
<rdfs:Class rdf:ID="animal" />  
  
<rdfs:Class rdf:ID="horse">  
  <rdfs:subClassOf rdf:resource="#animal"/>  
</rdfs:Class>  
  
</rdf:RDF>*

*That's it!*

*The Dublin Core*

*The Dublin Core Metadata Initiative (DCMI) has created some predefined properties for describing documents.*

*RDF is metadata (data about data). RDF is used to describe information resources.*

*The Dublin Core is a set of predefined properties for describing documents.*

*The first Dublin Core properties were defined at the****Metadata Workshop in Dublin, Ohio****in 1995 and is currently maintained by the*[*Dublin Core Metadata Initiative*](http://dublincore.org/)*.*

|  |  |
| --- | --- |
| ***Property*** | ***Definition*** |
| *Contributor* | *An entity responsible for making contributions to the content of the resource* |
| *Coverage* | *The extent or scope of the content of the resource* |
| *Creator* | *An entity primarily responsible for making the content of the resource* |
| *Format* | *The physical or digital manifestation of the resource* |
| *Date* | *A date of an event in the lifecycle of the resource* |
| *Description* | *An account of the content of the resource* |
| *Identifier* | *An unambiguous reference to the resource within a given context* |
| *Language* | *A language of the intellectual content of the resource* |
| *Publisher* | *An entity responsible for making the resource available* |
| *Relation* | *A reference to a related resource* |
| *Rights* | *Information about rights held in and over the resource* |
| *Source* | *A Reference to a resource from which the present resource is derived* |
| *Subject* | *A topic of the content of the resource* |
| *Title* | *A name given to the resource* |
| *Type* | *The nature or genre of the content of the resource* |

*// аспекты трэпр. Цікавая ініцыятыва. //*

*A quick look at the table above indicates that RDF is ideal for representing Dublin Core information.*

*RDF Example*

*The following example demonstrates the use of some of the Dublin Core properties in an RDF document:*

*<?xml version="1.0"?>  
  
<rdf:RDF  
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"  
xmlns:dc= "http://purl.org/dc/elements/1.1/">  
  
<rdf:Description rdf:about="https://www.w3schools.com">  
  <dc:description>W3Schools - Free tutorials</dc:description>  
  <dc:publisher>Refsnes Data as</dc:publisher>  
  <dc:date>2008-09-01</dc:date>  
  <dc:type>Web Development</dc:type>  
  <dc:format>text/html</dc:format>  
  <dc:language>en</dc:language>  
</rdf:Description>  
  
</rdf:RDF>*

*RDF Reference*

*The RDF namespace (xmlns:rdf) is:*[*http://www.w3.org/1999/02/22-rdf-syntax-ns#*](https://www.w3schools.com/xml/rdf-syntax-ns.xml)

*The RDFS namespace (xmlns:rdfs ) is:*[*http://www.w3.org/2000/01/rdf-schema#*](https://www.w3schools.com/xml/rdf-schema.xml)

*The recommended file extension for RDF files is****.rdf****. However, the extension****.xml****is often used to provide compatibility with old xml parsers.*

*The MIME type should be****"application/rdf+xml"****.*

*RDFS / RDF Classes*

|  |  |  |
| --- | --- | --- |
| ***Element*** | ***Class of*** | ***Subclass of*** |
| *rdfs:Class* | *All classes* |  |
| *rdfs:Datatype* | *Data types* | *Class* |
| *rdfs:Resource* | *All resources* | *Class* |
| *rdfs:Container* | *Containers* | *Resource* |
| *rdfs:Literal* | *Literal values (text and numbers)* | *Resource* |
| *rdf:List* | *Lists* | *Resource* |
| *rdf:Property* | *Properties* | *Resource* |
| *rdf:Statement* | *Statements* | *Resource* |
| *rdf:Alt* | *Containers of alternatives* | *Container* |
| *rdf:Bag* | *Unordered containers* | *Container* |
| *rdf:Seq* | *Ordered containers* | *Container* |
| *rdfs:ContainerMembershipProperty* | *Container membership properties* | *Property* |
| *rdf:XMLLiteral* | *XML literal values* | *Literal* |

*// intbu about using of rdfs //*

*RDFS / RDF Properties*

|  |  |  |  |
| --- | --- | --- | --- |
| ***Element*** | ***Domain*** | ***Range*** | ***Description*** |
| *rdfs:domain* | *Property* | *Class* | *The domain of the resource* |
| *rdfs:range* | *Property* | *Class* | *The range of the resource* |
| *rdfs:subPropertyOf* | *Property* | *Property* | *The property is a sub property of a property* |
| *rdfs:subClassOf* | *Class* | *Class* | *The resource is a subclass of a class* |
| *rdfs:comment* | *Resource* | *Literal* | *The human readable description of the resource* |
| *rdfs:label* | *Resource* | *Literal* | *The human readable label (name)  of the resource* |
| *rdfs:isDefinedBy* | *Resource* | *Resource* | *The definition of the resource* |
| *rdfs:seeAlso* | *Resource* | *Resource* | *The additional information about the resource* |
| *rdfs:member* | *Resource* | *Resource* | *The member of the resource* |
| *rdf:first* | *List* | *Resource* |  |
| *rdf:rest* | *List* | *List* |  |
| *rdf:subject* | *Statement* | *Resource* | *The subject of the resource in an RDF Statement* |
| *rdf:predicate* | *Statement* | *Resource* | *The predicate of the resource in an RDF Statement* |
| *rdf:object* | *Statement* | *Resource* | *The object of the resource in an RDF Statement* |
| *rdf:value* | *Resource* | *Resource* | *The property used for values* |
| *rdf:type* | *Resource* | *Class* | *The resource is an instance of a class* |

*RDF Attributes*

|  |  |
| --- | --- |
| ***Attribute*** | ***Description*** |
| *rdf:about* | *Defines the resource being described* |
| *rdf:Description* | *Container for the description of a resource* |
| *rdf:resource* | *Defines a resource to identify a property* |
| *rdf:datatype* | *Defines the data type of an element* |
| *rdf:ID* | *Defines the ID of an element* |
| *rdf:li* | *Defines a list* |
| *rdf:\_n* | *Defines a node* |
| *rdf:nodeID* | *Defines the ID of an element node* |
| *rdf:parseType* | *Defines how an element should be parsed* |
| *rdf:RDF* | *The root of an RDF document* |
| *xml:base* | *Defines the XML base* |
| *xml:lang* | *Defines the language of the element content* |

*// потым варта будзе лепей адчуць прызначэнне і пользу дадзенай тэхналогіі… А таксама павыкарыстоўваць яе //*

*// ці з’яўляецца атрыбутамі ўсё тое, што пералічана ў табліцы? //*

[*https://www.w3schools.com/xml/xml\_rss.asp*](https://www.w3schools.com/xml/xml_rss.asp)

*XML RSS*

*With RSS it is possible to distribute up-to-date web content from one web site to thousands of other web sites around the world.*

*RSS allows fast browsing for news and updates.*

*RSS Document Example*

*<?xml version="1.0" encoding="UTF-8" ?>  
<rss version="2.0">  
  
<channel>  
  <title>W3Schools Home Page</title>  
  <link>https://www.w3schools.com</link>  
  <description>Free web building tutorials</description>  
  <item>  
    <title>RSS Tutorial</title>  
    <link>https://www.w3schools.com/xml/xml\_rss.asp</link>  
    <description>New RSS tutorial on W3Schools</description>  
  </item>  
  <item>  
    <title>XML Tutorial</title>  
    <link>https://www.w3schools.com/xml</link>  
    <description>New XML tutorial on W3Schools</description>  
  </item>  
</channel>  
  
</rss>*

*What is RSS?*

* *RSS stands for****R****eally****S****imple****S****yndication*
* *RSS allows you to syndicate your site content*
* *RSS defines an easy way to share and view headlines and content*
* *RSS files can be automatically updated*
* *RSS allows personalized views for different sites*
* *RSS is written in XML*

*Why use RSS?*

*RSS was designed to show selected data.*

*Without RSS, users will have to check your site daily for new updates. This may be too time-consuming for many users. With an RSS feed (RSS is often called a News feed or RSS feed) they can check your site faster using an RSS aggregator (a site or program that gathers and sorts out RSS feeds).*

*Since RSS data is small and fast-loading, it can easily be used with services like cell phones or PDA's.*

*Web-rings with similar information can easily share data on their web sites to make them better and more useful.*

*Who Should use RSS?*

*Webmasters who seldom update their web sites do not need RSS!*

*RSS is useful for web sites that are updated frequently, like:*

* *News sites - Lists news with title, date and descriptions*
* *Companies - Lists news and new products*
* *Calendars - Lists upcoming events and important days*
* *Site changes - Lists changed pages or new pages*

*Benefits of RSS*

*Here are some benefits of using RSS:*

***Choose your news***

*With RSS you can choose to view the news you want, the news that interest you and are relevant to your work.*

***Remove unwanted information***

*With RSS you can (finally) separate wanted information from unwanted information (spam)!*

***Increase your site traffic***

*With RSS you can create your own news channel, and publish it to the Internet!*

*The History of RSS*

* *1997 - Dave Winer at UserLand develops scriptingNews. RSS was born*
* *1999 - Netscape develops RSS 0.90 (which supported scriptingNews)*
* *1999 - Dave Winer develops scriptingNews 2.0b1 (which included RSS 0.90 features)*
* *1999 - Netscape develops RSS 0.91 (which included most features from scriptingNews 2.0b1)*
* *1999 - UserLand gets rid of scriptingNews and uses only RSS 0.91*
* *1999 - Netscape stops their RSS development*
* *2000 - UserLand releases the official RSS 0.91 specification*
* *2000 - O'Reilly develops RSS 1.0. This format uses RDF and namespaces.*
* *2000 - Dave Winer at UserLand develops RSS 0.92*
* *2002 - Dave Winer develops RSS 2.0 after leaving UserLand*
* *2003 - The official RSS 2.0 specification is released*

*RSS 1.0 is the only version that was developed using the W3C RDF (Resource Description Framework) standard.*

*The idea behind RDF was to help create a Semantic Web (*[*https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D0%BC%D0%B0%D0%BD%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F\_%D0%BF%D0%B0%D1%83%D1%82%D0%B8%D0%BD%D0%B0*](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D0%BC%D0%B0%D0%BD%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D0%BF%D0%B0%D1%83%D1%82%D0%B8%D0%BD%D0%B0) *). However, this does not matter too much for ordinary users, but by using web standards it will be easier for persons and applications to exchange data.*

*What RSS Version Should I Use?*

*RSS 0.91 and RSS 2.0 are easier to understand than RSS 1.0. Our tutorial is based on RSS 2.0.*

*The syntax rules of RSS 2.0 are very simple and very strict.*

*Is RSS a Web Standard?*

*There is no official standard for RSS.*

* *About 50 % of all RSS feeds use RSS 0.91*
* *About 25 % use RSS 1.0*
* *The last 25 % is split between RSS 0.9x versions and RSS 2.0*

*How RSS Works*

*RSS is used to share content between websites.*

*With RSS, you register your content with companies called aggregators.*

*So, to be a part of it: First, create an RSS document and save it with an .xml extension. Then, upload the file to your website. Next, register with an RSS aggregator. Each day the aggregator searches the registered websites for RSS documents, verifies the link, and displays information about the feed so clients can link to documents that interests them.*

*// без неабходнасці непасрэдна заходзіць на ўсе веб-сайты? Зручна! //*

*// агрэгатар паведамляе падпісчыкаў аб новенькіх навінах! //*

***Tip:****Read our RSS Publishing chapter to view free RSS aggregation services.*

*RSS Example*

*RSS documents use a self-describing and simple syntax.*

*Here is a simple RSS document:*

*<?xml version="1.0" encoding="UTF-8" ?>  
<rss version="2.0">  
  
<channel>  
  <title>W3Schools Home Page</title>  
  <link>https://www.w3schools.com</link>  
  <description>Free web building tutorials</description>  
  <item>  
    <title>RSS Tutorial</title>  
    <link>https://www.w3schools.com/xml/xml\_rss.asp</link>  
    <description>New RSS tutorial on W3Schools</description>  
  </item>  
  <item>  
    <title>XML Tutorial</title>  
    <link>https://www.w3schools.com/xml</link>  
    <description>New XML tutorial on W3Schools</description>  
  </item>  
</channel>  
  
</rss>*

*The first line in the document - the XML declaration - defines the XML version and the character encoding used in the document. In this case the document conforms to the 1.0 specification of XML and uses the UTF-8 character set.*

*The next line is the RSS declaration which identifies that this is an RSS document (in this case, RSS version 2.0).*

*The next line contains the <channel> element. This element is used to describe the RSS feed.*

*The <channel> element has three required child elements:*

* *<title> - Defines the title of the channel (e.g. W3Schools Home Page)*
* *<link> - Defines the hyperlink to the channel (e.g. https://www.w3schools.com)*
* *<description> - Describes the channel (e.g. Free web building tutorials)*

*Each <channel> element can have one or more <item> elements.*

*Each <item> element defines an article or "story" in the RSS feed.*

*The <item> element has three required child elements:*

* *<title> - Defines the title of the item (e.g. RSS Tutorial)*
* *<link> - Defines the hyperlink to the item (e.g. https://www.w3schools.com/xml/xml\_rss.asp)*
* *<description> - Describes the item (e.g. New RSS tutorial on W3Schools)*

*Finally, the two last lines close the <channel> and <rss> elements.*

*Comments in RSS*

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

*<!-- This is an RSS comment -->*

*RSS is Written in XML*

*Because RSS is XML, keep in mind that:*

* *All elements must have a closing tag*
* *Elements are case sensitive*
* *Elements must be properly nested*
* *Attribute values must always be quoted*

*The RSS <channel> Element*

*The RSS <channel> element describes the RSS feed.*

*Look at the following RSS document:*

*<?xml version="1.0" encoding="UTF-8" ?>  
<rss version="2.0">  
  
<channel>  
  <title>W3Schools Home Page</title>  
  <link>https://www.w3schools.com</link>  
  <description>Free web building tutorials</description>  
  <item>  
    <title>RSS Tutorial</title>  
    <link>https://www.w3schools.com/xml/xml\_rss.asp</link>  
    <description>New RSS tutorial on W3Schools</description>  
  </item>  
</channel>  
  
</rss>*

*As mentioned before, the <channel> element describes the RSS feed, and has three required child elements: // стужка навінаў //*

* *<title> - Defines the title of the channel (e.g. W3Schools Home Page)*
* *<link> - Defines the hyperlink to the channel (e.g. https://www.w3schools.com)*
* *<description> - Describes the channel (e.g. Free web building tutorials)*

*The <channel> element usually contains one or more <item> elements. Each <item> element defines an article or "story" in the RSS feed.*

*Furthermore, there are several optional child elements of <channel>. We will explain the most important ones below.*

*The <category> Element*

*The <category> child element is used to specify a category for your feed.*

*The <category> element makes it possible for RSS aggregators to group sites based on category.*

*The category for the RSS document above could be:*

*<category>Web development</category>*

*The <copyright> Element*

*The <copyright> child element notifies about copyrighted material.*

*The copyright for the RSS document above could be:*

*<copyright>2006 Refsnes Data as. All rights reserved.</copyright>*

*The <image> Element*

*The <image> child element allows an image to be displayed when aggregators present a feed.*

*The <image> element has three required child elements:*

* *<url> - Defines the URL to the image*
* *<title> - Defines the text to display if the image could not be shown*
* *<link> - Defines the hyperlink to the website that offers the channel*

*The image for the RSS document above could be:*

*<image>  
  <url>https://www.w3schools.com/images/logo.gif</url>  
  <title>W3Schools.com</title>  
  <link>https://www.w3schools.com</link>  
</image>*

*The <language> Element*

*The <language> child element is used to specify the language used to write your document.*

*The <language> element makes it possible for RSS aggregators to group sites based on language.*

*The language for the RSS document above could be:*

*<language>en-us</language>*

*The <item> Element*

*Each <item> element defines an article or "story" in an RSS feed.*

*Look at the following RSS document:*

*<?xml version="1.0" encoding="UTF-8" ?>  
<rss version="2.0">  
  
<channel>  
  <title>W3Schools Home Page</title>  
  <link>https://www.w3schools.com</link>  
  <description>Free web building tutorials</description>  
  <item>  
    <title>RSS Tutorial</title>  
    <link>https://www.w3schools.com/xml/xml\_rss.asp</link>  
    <description>New RSS tutorial on W3Schools</description>  
  </item>  
</channel>  
  
</rss>*

*As mentioned before, each <item> element defines an article or "story" in the RSS feed.*

*The <item> element has three required child elements:*

* *<title> - Defines the title of the item (e.g. RSS Tutorial)*
* *<link> - Defines the hyperlink to the item (e.g. https://www.w3schools.com/xml/xml\_rss.asp)*
* *<description> - Describes the item (e.g. New RSS tutorial on W3Schools)*

*Furthermore, there are several optional child elements of <item>. We will explain the most important ones below.*

*The <author> Element*

*The <author> child element is used to specify the e-mail address of the author of an item.*

***Note:****To prevent spam e-mails, some developers do not include the <author> element.*

*The author of the item in the RSS document above could be:*

*<author>hege@refsnesdata.no</author>*

*The <comments> Element*

*The <comments> child element allows an item to link to comments about that item.*

*A comment of the item in the RSS document above could be:*

*<comments>https://www.w3schools.com/comments</comments>*

*The <enclosure> Element*

*The <enclosure> child element allows a media-file to be included with an item.*

*The <enclosure> element has three required attributes:*

* *url - Defines the URL to the media file*
* *length - Defines the length (in bytes) of the media file*
* *type - Defines the type of media file*

*A media-file included in the item in the RSS document above could be:*

*<enclosure url="https://www.w3schools.com/xml/rss.mp3"  
length="5000" type="audio/mpeg" />*

*Get Your RSS Feed Up On The Web*

*Having an RSS document is not useful if other people cannot reach it.*

*Now it's time to get your RSS file up on the web. Here are the steps:*

*1. Name your RSS file. Notice that the file must have an .xml extension.*

*2. Validate your RSS file (a good validator can be found at*[*http://www.feedvalidator.org*](http://www.feedvalidator.org/)*).*

*3. Upload the RSS file to your web directory on your web server.*

*4. Copy the little orange RSS Logo or XML Logo button to your web directory.*

*5. Put the little orange "RSS" or "XML" button on the page where you will offer RSS to the world (e.g. on your home page). Then add a link to the button that links to the RSS file. The code will look something like this:  
<a href="https://www.w3schools.com/xml/myfirstrss.xml">  
<img src="https://www.w3schools.com/xml/pic\_rss.gif" width="36" height="14">  
</a>.*

*6. Submit your RSS feed to the RSS Feed Directories (you can Google or Yahoo for "RSS Feed Directories"). Note! The URL to your feed is not your home page, it is the URL to your feed, like "https://www.w3schools.com/xml/myfirstrss.xml". Here is a free RSS aggregation service:*

* [*Newsisfree*](http://www.newsisfree.com/)*:*[*Register here*](http://www.newsisfree.com/user/new/)

*7. Register your feed with the major search engines:*

* *Google -*[*http://www.google.com/submityourcontent/website-owner*](http://www.google.com/submityourcontent/website-owner)
* *Bing -*[*http://www.bing.com/toolbox/submit-site-url*](http://www.bing.com/toolbox/submit-site-url)

*8. Update your feed - After registering your RSS feed, you must make sure that you update your content frequently and that your RSS feed is constantly available.*

*Can I Manage my RSS Feed Myself?*

*The best way to ensure your RSS feed works the way you want, is to manage it yourself.*

*However, this can be very time consuming, especially for pages with lot of updates.*

*An alternative is to use a third-party automated RSS.*

*// а ў чым заключаецца кіраванне стужкай? //*

*Automated RSS*

*For users who only need an RSS feed for their personal website, some of the most popular blog (Web Log) managers that offer built-in RSS services are:*

* [*Wordpress*](http://wordpress.org/)
* [*Blogger*](http://www.blogger.com/)
* [*Radio*](http://radio.userland.com/)

*RSS Readers*

*An RSS Reader is used to read RSS Feeds!*

*RSS readers are available for many different devices and OS.*

*There are a lot of different RSS readers. Some work as web services, and some are limited to windows (or Mac, PDA or UNIX):*

* [*QuiteRSS*](http://quiterss.org/en)*- FREE! QuiteRSS is an open-source, cross-platform RSS/Atom news reader. It is versatile, and has a full set of options. QuietRSS has a rich set of social sharing options (Email/Twitter/Facebook/.../Pocket/Etc). QuietRSS is fast starting, and navigation is quick*
* [*FeedReader*](http://feedreader.com/)*- FREE! Simple, straightforward feed reader that easily handles large number of feeds. Has the essential options (not a lot of confusing ones). Does not require Java. Import or export OPML files. Option to open links in an external browser*
* [*RssReader*](http://www.rssreader.com/)*- FREE! Windows-based RSS reader. Supports RSS versions 0.9x, 1.0 and 2.0 and Atom 0.1, 0.2 and 0.3*
* [*blogbotrss*](http://sourceforge.net/projects/blogbotrss/)*- FREE! An RSS reader plug-in for Internet Explorer and Microsoft Outlook*

***Tip:****Most browsers have a built-in RSS Reader. If you go to a web site that offers RSS feeds, you will see an RSS icon in the address bar, or toolbar. Click on the icon to view a list of the different feeds. Choose the feed you want to read.*

*// RSS – цікавая тэхналогія. Але мне варта засвоіць яе на 7ураз //*

*I have an RSS Reader. Now what?*

*Click on the little RSS Logo or XML Logo button next to the RSS feed you want to read. Copy The URL you get in the browser window and paste it in your RSS reader.*

*// потым варта будзе лепей разабрацца, бо ні на мэйле, ні на тут.бае я такога не заўважыў //*

*RSS Examples*

*These examples demonstrate RSS using our RSS reader to view the results.*

***RSS <channel> Element***

[*Setting the required channel elements (<title>, <link>, and <description>)*](https://www.w3schools.com/xml/tryrss.asp?filename=rss_ex_title_link_description)[*Specify a category for the RSS*](https://www.w3schools.com/xml/tryrss.asp?filename=rss_ex_category)[*Specify the program used to generate the RSS*](https://www.w3schools.com/xml/tryrss.asp?filename=rss_ex_generator)[*Add an image to the RSS*](https://www.w3schools.com/xml/tryrss.asp?filename=rss_ex_image)[*Specify the language of the RSS*](https://www.w3schools.com/xml/tryrss.asp?filename=rss_ex_language)[*Add an text input field to the RSS*](https://www.w3schools.com/xml/tryrss.asp?filename=rss_ex_textinput)[*Specify days that the RSS should not be updated*](https://www.w3schools.com/xml/tryrss.asp?filename=rss_ex_skipdays)

***RSS <item> element***

[*Setting the required item elements (<title>, <link>, and <description>)*](https://www.w3schools.com/xml/tryrss.asp?filename=rss_ex_title_link_description_item)[*Add a link to comments about the RSS item*](https://www.w3schools.com/xml/tryrss.asp?filename=rss_ex_comments)[*Add a media file to the RSS item*](https://www.w3schools.com/xml/tryrss.asp?filename=rss_ex_enclosure)[*Specify a unique identifier for the item*](https://www.w3schools.com/xml/tryrss.asp?filename=rss_ex_guid)[*Specify the publication date for the RSS item*](https://www.w3schools.com/xml/tryrss.asp?filename=rss_ex_pubdate_item)[*Specify a third-party source for the RSS item*](https://www.w3schools.com/xml/tryrss.asp?filename=rss_ex_source)

*// пакуль не чытаў //*

*RSS Reference*

*RSS <channel> Element*

*The links in the "Element" column point to more information about each specific element.*

|  |  |
| --- | --- |
| ***Element*** | ***Description*** |
| [*<category>*](https://www.w3schools.com/xml/rss_tag_category_channel.asp) | *Optional. Defines one or more categories for the feed* |
| [*<cloud>*](https://www.w3schools.com/xml/rss_tag_cloud.asp) | *Optional. Register processes to be notified immediately of updates of the feed* |
| [*<copyright>*](https://www.w3schools.com/xml/rss_tag_copyright.asp) | *Optional. Notifies about copyrighted material* |
| [*<description>*](https://www.w3schools.com/xml/rss_tag_title_link_description_channel.asp) | *Required. Describes the channel* |
| [*<docs>*](https://www.w3schools.com/xml/rss_tag_docs.asp) | *Optional. Specifies a URL to the documentation of the format used in the feed* |
| [*<generator>*](https://www.w3schools.com/xml/rss_tag_generator.asp) | *Optional. Specifies the program used to generate the feed* |
| [*<image>*](https://www.w3schools.com/xml/rss_tag_image.asp) | *Optional. Allows an image to be displayed when aggregators present a feed* |
| [*<language>*](https://www.w3schools.com/xml/rss_tag_language.asp) | *Optional. Specifies the language the feed is written in* |
| [*<lastBuildDate>*](https://www.w3schools.com/xml/rss_tag_lastbuilddate.asp) | *Optional. Defines the last-modified date of the content of the feed* |
| [*<link>*](https://www.w3schools.com/xml/rss_tag_title_link_description_channel.asp) | *Required. Defines the hyperlink to the channel* |
| [*<managingEditor>*](https://www.w3schools.com/xml/rss_tag_managingeditor.asp) | *Optional. Defines the e-mail address to the editor of the content of the feed* |
| [*<pubDate>*](https://www.w3schools.com/xml/rss_tag_pubdate.asp) | *Optional. Defines the last publication date for the content of the feed* |
| *<rating>* | *Optional. The PICS rating of the feed* |
| [*<skipDays>*](https://www.w3schools.com/xml/rss_tag_skipDays.asp) | *Optional. Specifies the days where aggregators should skip updating the feed* |
| [*<skipHours>*](https://www.w3schools.com/xml/rss_tag_skipHours.asp) | *Optional. Specifies the hours where aggregators should skip updating the feed* |
| [*<textInput>*](https://www.w3schools.com/xml/rss_tag_textinput.asp) | *Optional. Specifies a text input field that should be displayed with the feed* |
| [*<title>*](https://www.w3schools.com/xml/rss_tag_title_link_description_channel.asp) | *Required. Defines the title of the channel* |
| [*<ttl>*](https://www.w3schools.com/xml/rss_tag_ttl.asp) | *Optional. Specifies the number of minutes the feed can stay cached before refreshing it from the source* |
| [*<webMaster>*](https://www.w3schools.com/xml/rss_tag_webmaster.asp) | *Optional. Defines the e-mail address to the webmaster of the feed* |

*RSS <item> Element*

|  |  |
| --- | --- |
| ***Element*** | ***Description*** |
| [*<author>*](https://www.w3schools.com/xml/rss_tag_author.asp) | *Optional. Specifies the e-mail address to the author of the item* |
| [*<category>*](https://www.w3schools.com/xml/rss_tag_category_item.asp) | *Optional. Defines one or more categories the item belongs to* |
| [*<comments>*](https://www.w3schools.com/xml/rss_tag_comments.asp) | *Optional. Allows an item to link to comments about that item* |
| [*<description>*](https://www.w3schools.com/xml/rss_tag_title_link_description_item.asp) | *Required. Describes the item* |
| [*<enclosure>*](https://www.w3schools.com/xml/rss_tag_enclosure.asp) | *Optional. Allows a media file to be included with the item* |
| [*<guid>*](https://www.w3schools.com/xml/rss_tag_guid.asp) | *Optional. Defines a unique identifier for the item* |
| [*<link>*](https://www.w3schools.com/xml/rss_tag_title_link_description_item.asp) | *Required. Defines the hyperlink to the item* |
| [*<pubDate>*](https://www.w3schools.com/xml/rss_tag_pubdate_item.asp) | *Optional. Defines the last-publication date for the item* |
| [*<source>*](https://www.w3schools.com/xml/rss_tag_source.asp) | *Optional. Specifies a third-party source for the item* |
| [*<title>*](https://www.w3schools.com/xml/rss_tag_title_link_description_item.asp) | *Required. Defines the title of the item* |