XML

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".
- DTD stands for Document Type Definition.
- A DTD defines the structure and the legal elements and attributes of an XML document.

Valid XML Documents

```
<?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>
```

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

The DOCTYPE declaration above contains a reference to a DTD file.

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 parseable character data.

XML DTD

When to Use a DTD?

- 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.

When NOT to Use a DTD?

- XML does not require a DTD.
- When you are experimenting with XML, avoid it
- 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 DTD. Otherwise, your software might stop working because of validation errors.

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 is an XML-based alternative to DTD

Valid XML Documents

```
<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 SCHEMA

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 TO USE XML SCHEMA

With XML Schema,

- your XML files can carry a description of its own format.
- independent groups of people can agree on a standard for interchanging data.
- you can verify data.

XML Schemas Support Data Types

One of the greatest strengths 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

WHY TO USE XML SCHEMA

XML Schemas use XML Syntax

- 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

XSL – eXtensible Stylesheet Language

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

XSLT

- stands for XSL Transformations
- important part of XSL
- transforms an XML document into another XML document
- uses XPath to navigate in XML documents
- XSLT is a W3C Recommendation

XSL Transformation

- 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.
- 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 Example

```
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
```

```
<xsl:transform version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
```

- The root element that declares the document to be an XSL style sheet is <xsl:stylesheet> or <xsl:transform>.
- <xsl:stylesheet> and <xsl:transform> are completely synonymous and either can be used

- 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".

XSLT Example

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>
```

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
▼<catalog>
  ▼<cd>
      <title>Empire Burlesque</title>
      <artist>Bob Dylan</artist>
      <country>USA</country>
      <company>Columbia</company>
      <price>10.90</price>
      <year>1985</year>
    </cd>
  ▼<cd>
      <title>Hide your heart</title>
      <artist>Bonnie Tyler</artist>
      <country>UK</country>
      <company>CBS Records</company>
      <price>9.90</price>
      <year>1988</year>
    </cd>
  ▼<cd>
      <title>Greatest Hits</title>
      <artist>Dolly Parton</artist>
      <country>USA</country>
      <company>RCA</company>
      <price>9.90</price>
      <year>1982</year>
    </cd>
  ▼<cd>
      <title>Still got the blues</title>
      <artist>Gary Moore</artist>
```

XSLT 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>
```

Create an XSL Stylesheet

XSLT Example

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-
stylesheet type="text/xsl" href="cdcatalog.xsl"?>
<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>
```

Link the XSL Style Sheet to the XML Document

Add the XSL style sheet reference to your XML document ("cdcatalog.xml"):

XSLT Example

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-
stylesheet type="text/xsl" href="cdcatalog.xsl"?>
<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>
```

Result

My CD Collection

Title	Artist
Empire Burlesque	Bob Dylan
Hide your heart	Bonnie Tyler
Greatest Hits	Dolly Parton
Still got the blues	Gary Moore
Eros	Eros Ramazzotti
One night only	Bee Gees
Sylvias Mother	Dr.Hook
Maggie May	Rod Stewart
Romanza	Andrea Bocelli
When a man loves a woman	Percy Sledge
Black angel	Savage Rose
1999 Grammy Nominees	Many
For the good times	Kenny Rogers
Big Willie style	Will Smith
Tupelo Honey	Van Morrison
Soulsville	Jorn Hoel
The very best of	Cat Stevens
Stop	Sam Brown
Bridge of Spies	T`Pau
Private Dancer	Tina Turner
Midt om natten	Kim Larsen
Pavarotti Gala Concert	Luciano Pavarotti
The dock of the bay	Otis Redding
Picture book	Simply Red
Red	The Communards
Unchain my heart	Joe Cocker

XML Example

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

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

The tags in the example above (like <to> and <from>) are not defined in any XML standard.

These tags are "invented" by the author of the XML document.

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

XML is Extensible

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

```
<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>
```

- Most XML applications will work as expected even if new data is added (or removed).
- Imagine an application designed to display the original version of note.xml (<to> <from> <heading> <body>).
- Then imagine a newer version of note.xml with added <date> and <hour> elements, and a removed <heading>.
- The way XML is constructed, older version of the application can still work

XML SIMPLIFIES THINGS

Data Sharing

- 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.

Data Transport

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

XML SIMPLIFIES THINGS

Platform sharing

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

Data Availability

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

HOW TO USE XML?

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.

HOW TO USE XML?

XML Separates Data from HTML

- When displaying data in HTML, you should not have to edit the HTML file when the data changes.
- 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.

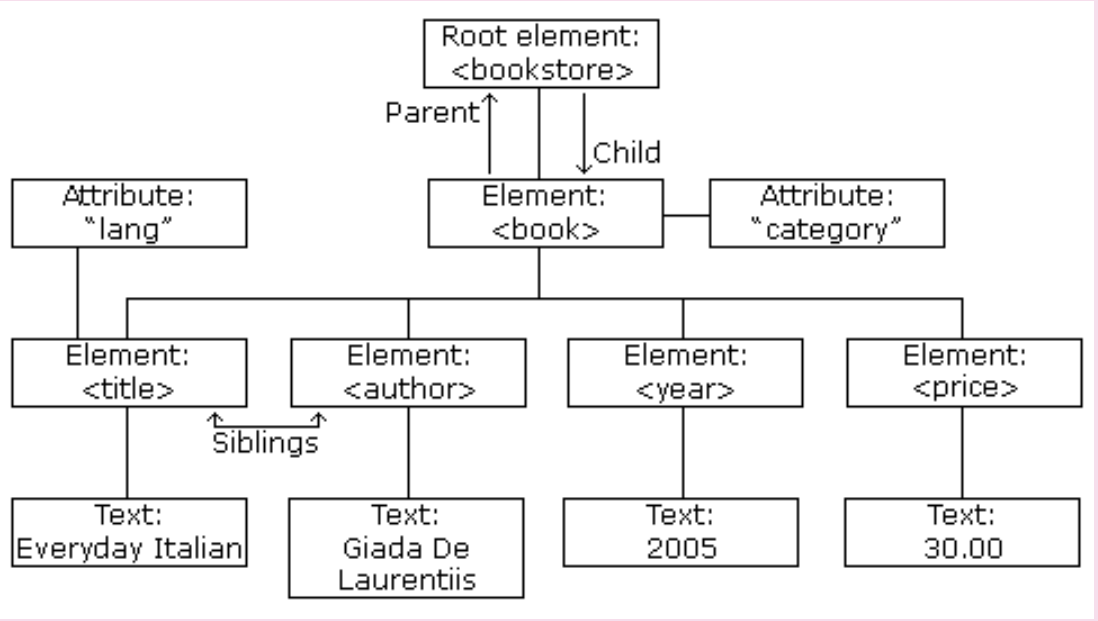
Books.xml

```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" cover="paperback">
    <title lang="en">Learning XML</title>
    <author>Erik T. Ray</author>
    <year>2003</year>
    <price>39.95</price>
  </book>

</bookstore>
```

Title	Author
Everyday Italian	Giada De Laurentiis
Harry Potter	J K. Rowling
XQuery Kick Start	James McGovern
Learning XML	Erik T. Ray

XML Tree

```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">Learning XML</title>
    <author>Erik T. Ray</author>
    <year>2003</year>
    <price>39.95</price>
  </book>
</bookstore>
```



XML Tree

```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">Learning XML</title>
    <author>Erik T. Ray</author>
    <year>2003</year>
    <price>39.95</price>
  </book>
</bookstore>
```

- 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.
- Parents have children. Children have parents. Siblings are children on the same level (brothers and sisters).
- All elements can have text content (Harry Potter) and attributes (category="cooking").

XML Syntax Rule

```
<?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">Learning XML</title>
    <author>Erik T. Ray</author>
    <year>2003</year>
    <price>39.95</price>
  </book>
</bookstore>
```

- XML document must have ROOT element
- XML Prolog: Optional, if exist then first statement
- All XML elements must have closing tag
- XML tags are case sensitive
- XML elements must be properly nested
- XML Attribute Values Must Always be Quoted
- White space is preserved in XML

Entity Reference

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

XML HTTPRequest

- All modern browsers have a built-in XMLHttpRequest object to request data from a server.
- The XMLHttpRequest object can be used to request data from a web server.

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

1. Update a web page without reloading the page
2. Request data from a server - after the page has loaded
3. Receive data from a server - after the page has loaded
4. Send data to a server - in the background

Sending an XMLHTTPRequest

```
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();
```

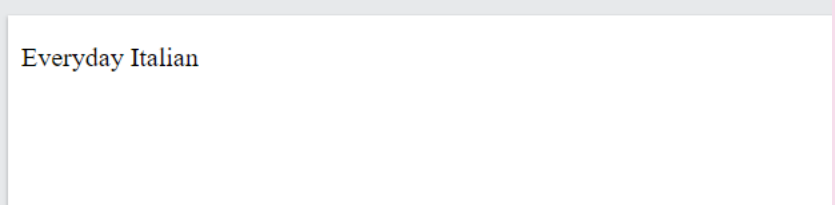
- The first line in the example above creates an **XMLHttpRequest** object
- The **onreadystatechange** property specifies a function to be executed every time the status of the XMLHttpRequest object changes
- When **readyState** property is 4 and the **status** property is 200, the response is ready
- The **responseText** property returns the server response as a text string.
- The text string can be used to update a web page

Sending an XMLHTTPRequest

```
<!DOCTYPE html>
<html>
<body>

<h2>Using the XMLHttpRequest Object</h2>

<div id="demo">
<button type="button" onclick="loadXMLDoc()">Change
Content</button>
</div>

<script>
function loadXMLDoc() {
  var xmlhttp;
  if (window.XMLHttpRequest) {
    xmlhttp = new XMLHttpRequest();
  } else {
    // code for older browsers
    xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
  }
  xmlhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("demo").innerHTML =
      this.responseText;
    }
  };
  xmlhttp.open("GET", "xmlhttp_info.txt", true);
  xmlhttp.send();
}
</script>

</body>
</html>
```

XML Parser

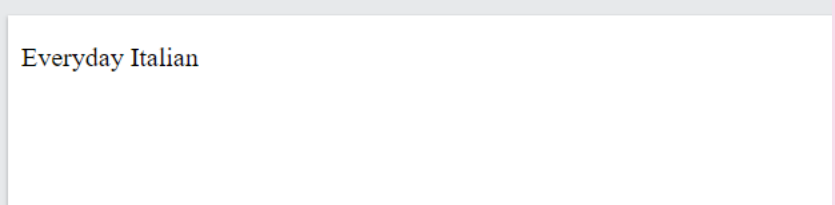
```
<html>
<body>

<p id="demo"></p>

<script>
var text, parser, xmlDoc;

// text string is defined
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>
```



- The XML DOM (Document Object Model) 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.

XML Parser

```
<html>
<body>

<p id="demo"></p>

<script>
var text, parser, xmlDoc;

// text string is defined
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>
```



- All XML elements can be accessed through the XML DOM.
- 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:

XPath

- is a major element in the XSLT standard.
- can be used to navigate through elements and attributes in an XML document.
- is a syntax for defining parts of an XML document
- uses path expressions to navigate in XML documents
- contains a library of standard functions
- is a major element in XSLT and in XQuery
- is a W3C recommendation

Xpath Path Expression

- 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 Path Expression

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