

Web Development

XML & XHTML

Content



1. XML and XHTML Overview

2. XML Components

3. DTD & XML Schema

4. XML Validation

5. XML Applications

1.1. XML (eXtensible Markup Language)

- ◆ A new standard by W3C, derived from SGML
- ◆ EXtensible Markup Language (XML) is a meta-language that describes the content of the document (self-describing data)
Java = Portable Programs; XML = Portable Data
- ◆ XML does not specify the tag set or grammar of the language
 - Tag Set – markup tags that have meaning to a language processor
 - Grammar – defines correct usage of a language's tag

1.1. XML (2)

◆ Applications of XML

- Media for data interchange
 - ◆ A better alternative to proprietary data formats
- B2B transactions on the Web
 - ◆ Electronic business orders (ebXML)
 - ◆ Financial Exchange (IFX)
 - ◆ Messaging exchange (SOAP)

```
<?xml version="1.0" encoding="utf-8"?>
<recipe>
    <name>Iced Tea</name>
    <description>An iced tea that we serve everyday</description>
    <preparation>...</preparation>
</recipe>
```

1.2. XML vs. SGML

- ◆ SGML (Standard Generalized Markup Language)
 - ISO-standard meta-language
 - Powerful but very complex, suffers from lack of industry support
 - The basis for XML, first published in 1988
- ◆ XML (eXtensible Markup Language)
 - Simpler yet offers most of the power of SGML because it is also a meta-language
 - More likely to have broad industry support, because many companies and universities involved in development

1.3. XML vs. HTML

- ◆ Both based on SGML
 - XML is a subset of SGML
 - HTML is a markup language written in SGML
- ◆ XML fundamentally separates content (data and language) from presentation; HTML specifies the presentation
- ◆ HTML explicitly defines a set of legal tags as well as the grammar (intended meaning)
 - <TABLE> ... </TABLE>
- ◆ XML allows any tags or grammar to be used (hence, eXtensible)
 - <BOOK> ... </BOOK>

1.3. XML vs. HTML (2)

◆ HTML

- Not extensible – cannot customize
 - ◆ Cannot accommodate special needs (e.g. mathematics, chemical formulas)
 - ◆ Proprietary, vendor-specific tags to extends capabilities
- Only codes for display, not document structure, semantics or content

◆ XML

- Can define own markup language → Flexible
- Tagging/content separate from display
- Reflects structure and semantics of documents
→ better searching and navigation

1.4. XHTML

- History of HTML
 - HTML 1.0
 - Created by Tim Berners-Lee and submitted to IETF
 - HTML 2.0
 - RFC1866 in Nov. 1995
 - HTML 3.2
 - Jan. 1997
 - moved from IETF to W3C
 - HTML 4.0
 - Dec. 1997
 - HTML 4.01
 - Dec. 1999
 - HTML 5.0
 - 2008
 - HTML 5.1
 - 2016

HTML4.01

- ▶ HTML4.01 has three versions

- ▶ Strict

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"  
      "http://www.w3.org/TR/html4/strict.dtd">
```

- ▶ Transitional

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"  
      "http://www.w3.org/TR/html4/loose.dtd">
```

- ▶ Frameset

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN"  
      "http://www.w3.org/TR/html4/frameset.dtd">
```

XHTML1.0

- ▶ Reformulation of HTML4.01 in XML

- ▶ more strict syntax than HTML

- ▶ Three types of XHTML1.0

- ▶ Strict

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"  
      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

- ▶ Transitional

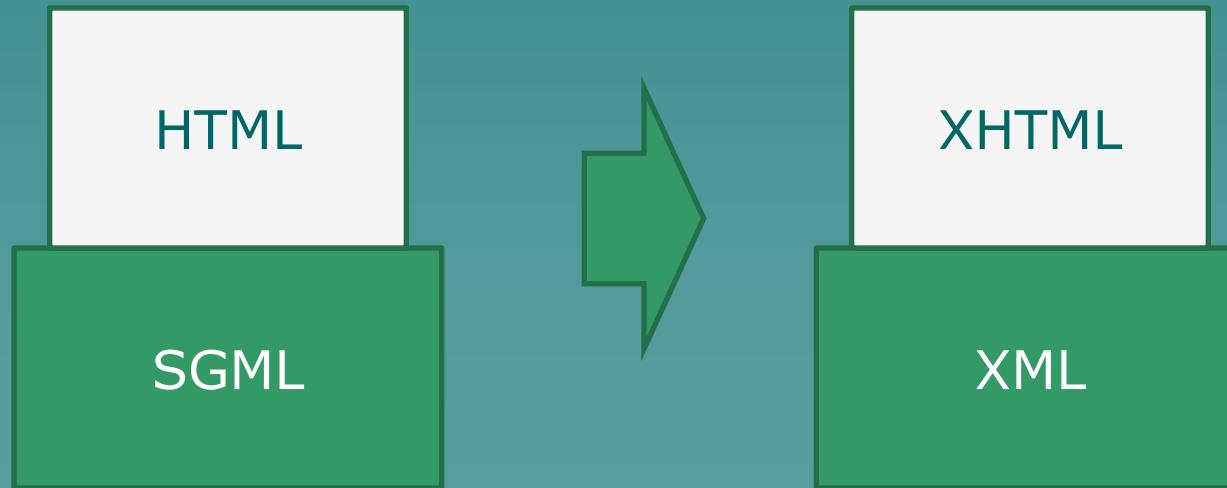
```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"  
      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

- ▶ Frameset

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"  
      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
```

HTML, XHTML and XML

- HTML is an SGML application
- XHTML is an XML application



1.5. XHTML Features

- ◆ Characters for a tag must be lower case
 - C <title>
 - I <TITLE>, <Title>
- ◆ Close tags must be needed
 - C <p>Para.</p>
 - I <p>Para<p>Next para
- ◆ An empty element needs " />" on the end
 - C
 - I

Example

```
  
  
<p lang="fr">Je  
levai la tête pour regarder les  
étoiles. Leur vue apaisa peu  
à peu les battements de mon  
coeur.</p>  
  
<p><cite class="title">Chroniques  
de l'oiseau à ressort</cite>  
- <cite class="author">Haruki  
Murakami</cite></p>
```

```
  
  
<p xml:lang="fr">Je  
levai la tête pour regarder les  
étoiles. Leur vue apaisa peu  
à peu les battements de mon  
coeur.</p>  
  
<p><cite class="title">Chroniques  
de l'oiseau à ressort</cite>  
- <cite class="author">Haruki  
Murakami</cite></p>
```

1.5. XHTML Features (2)

- ◆ An attribute element needs its value
 - C <select multiple="multiple" name="test">
 - I <**select multiple name="test"**>
- ◆ Attribute values must be quoted by the single quotation or the double quotation.
 - C <h1 id="title">Title</h1>
 - I <**h1 id=title**>Title</h1>

1.5. XHTML Features (3)

- ◆ XML Declaration is needed
 - <? xml version="1.0" encoding="utf-8" ?>
- ◆ xmlns attribute and xml:lang attribute
 - <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">

Example

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"DTD/xhtml11-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
<title> Strict DTD XHTML Example </title>
</head>
<body>
<p>
Please Choose a Day:
<br /><br />
<select name="day">
<option selected="selected">Monday</option>
<option>Tuesday</option>
<option>Wednesday</option>
</select>
</p>
</body>
</html>
```

Content

1. XML and XHTML Overview

→ 2. XML Components

3. DTD & XML Schema

4. XML Validation

5. XML Applications

2. XML Components

◆ Prolog

- Defines the xml version, entity definitions, and DOCTYPE

◆ Components of the document

- Tags and attributes
- CDATA (character data)
- Entities
- Processing instructions
- Comments

2.1. XML Prolog

- ◆ XML Files always start with a prolog
- ◆ Includes:
 - Declaration
 - Entities and DTD definitions

2.1.1. XML Declaration

◆ XML version and document encoding

```
<?xml version="1.0" encoding="ISO-8859-1"  
standalone="no"?>
```

- The version of XML is required
- The encoding identifies character set (default UTF-8)
- The value standalone identifies if an *external document is* referenced for DTD or entity definition

2.1.2. DOCTYPE Declaration

- ◆ Specifies the location of the DTD defining the syntax and structure of elements in the document
- ◆ Common forms:
 - `<!DOCTYPE root [DTD]>`
 - `<!DOCTYPE root SYSTEM URL>`
 - `<!DOCTYPE root PUBLIC FPI-identifier URL>`
- ◆ The root identifies the starting element (root element) of the document

2.1.2. DOCTYPE Declaration (2)

- ◆ The DTD can be external to the XML document, referenced by a SYSTEM or PUBLIC URL
 - SYSTEM URL refers to a private DTD
 - ◆ Located on the local file system or HTTP server
 - PUBLIC URL refers to a DTD intended for public use

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE authors SYSTEM "http://example.org/authors.dtd">
<authors>
...

```

Root element

URI Reference of DTD

SYSTEM or PUBLIC

DTD (Document Type Definition)

- ◆ A schema language for SGML and XML
 - Definitions of elements, attributes, entities
 - Content model: Tree structure by nested elements
- ◆ In authors.dtd on <http://example.org>:

```
<!DOCTYPE authors [  
  
    <!ELEMENT authors (name)*>  
    <!ELEMENT name (firstname, lastname)>  
    <!ELEMENT firstname (#PCDATA)>  
    <!ELEMENT lastname (#PCDATA)>  
  
]>
```

Simple XML Example

```
<?xml version="1.0"?>
<!DOCTYPE authors SYSTEM "http://example.org/authors.dtd">
<authors>
    <name>
        <firstname>Larry</firstname>
        <lastname>Brown</lastname>
    </name>
    <name>
        <firstname>Marty</firstname>
        <lastname>Hall</lastname>
    </name>
    ...
</authors>
```

Standalone XML document

```
<?xml version="1.0" standalone="yes"?>
<!DOCTYPE authors [
    <!ELEMENT authors (name)*>
    <!ELEMENT name (firstname, lastname)>
    <!ELEMENT firstname (#PCDATA)>
    <!ELEMENT lastname (#PCDATA)>
]>
<authors>
    <name>
        <firstname>James</firstname>
        <lastname>Gosling</lastname>
    </name>
    ...
</authors>
```

Specifying a PUBLIC DTD

<!DOCTYPE root PUBLIC *FPI-identifier URL*>

- ◆ The Formal Public Identifier (FPI) has four parts:
 - 1. Connection of DTD to a formal standard
 - ◆ - if defining yourself
 - ◆ + nonstandards body has approved the DTD
 - ◆ ISO if approved by formal standards committee
 - 2. Group responsible for the DTD
 - 3. Description and type of document
 - 4. Language used in the DTD
- ◆ E.g.

```
<!DOCTYPE Book PUBLIC "-//w3c//DTD XHTML 1.0 Transitional//EN"  
      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">  
<!DOCTYPE CWP PUBLIC "-//Prentice Hall//DTD Core Series  
      1.0//EN" "http://www.prenticehall.com/DTD/Core.dtd">
```

2.2. Component of the document

- ◆ Tags and attributes
- ◆ CDATA (character data)
- ◆ Entities
- ◆ Processing instructions
- ◆ Comments

2.2.1. XML Comment

◆ XML Comments

- The same as HTML comments
- <!-- This is an XML and HTML comment -->

2.2.2. Processing Instructions

- ◆ Application-specific instruction to the XML processor

```
<?processor-instruction?>
```

- ◆ Example

```
<?xml version="1.0" ?>
<?xml-stylesheet type="text/xml" href="orders.xsl" ?>
<orders>
    <order>
        <count>37</count>
        <price>49.99</price>
        <book>
            <isbn>0130897930</isbn>
            <title>Core Web Programming Second Edition</title>
            <authors>
                <author>Marty Hall</author>
                <author>Larry Brown</author>
            </authors>
        </book>
    </order>
</orders>
```

2.2.3. XML Root Element

- ◆ Required for XML-aware applications to recognize beginning and end of document
- ◆ Example

```
<?xml version="1.0" ?>
<book>
    <title>Core Web Programming</title>
    <contents>
        <chapter number="1"> Designing Web Pages with HTML
        </chapter>
        <chapter number="2"> Block-level Elements in HTML 4.0
        </chapter>
        <chapter number="3"> Text-level Elements in HTML 4.0
        </chapter>
        ...
    </contents>
</book>
```

2.2.4. XML Tags

- ◆ Tag names:
 - Case sensitive
 - Start with a letter or underscore
 - After first character, numbers, - and . are allowed
 - Cannot contain whitespaces
 - Avoid use of colon except for indicating namespaces
- ◆ For a well-formed XML documents
 - Every tag must have an end tag

```
<elementOne> ... </elementOne>  
<elementTwo />
```
 - All tags are completely nested (tag order cannot be mixed)

2.2.4. XML Tags (2)

- ◆ **Tags can also have attributes**

```
<message to="Gates@microsoft.com"  
from="Gosling@sun.com">  
  <priority/>  
  <text>We put the . in .com.  
      What did you do?  
  </text>  
</message>
```

2.2.5. XML Attributes

◆ Element Attributes

- Attributes provide metadata for the element
- Every attribute must be enclosed in "" with no commas in between
- Same naming conventions as elements

2.2.6. Document Entities

- ◆ Entities refer to a data item, typically text
 - General entity references start with & and end with ;
 - The entity reference is replaced by its true value when parsed
 - The characters < > & ' " require entity references to avoid conflicts with the XML application (parser)

< > & " '

- ◆ Entities are user definable

```
<?xml version="1.0" standalone="yes" ?>
<!DOCTYPE book [
    <!ELEMENT book (title)>
    <!ELEMENT title (#PCDATA)>
    <!ENTITY COPYRIGHT "2001, Prentice Hall">
]>
<book>
    <title>Core Web Programming, &COPYRIGHT;</title>
</book>
```

Content

1. XML and XHTML Overview

2. XML Components

3. DTD & XML Schema

4. XML Validation

5. XML Applications

Well-formed versus Valid

- ◆ An XML document can be *well-formed if it follows* basic syntax rules
- ◆ An XML document is *valid if its structure* matches a Document Type Definition (DTD) or an XML Schema

3.1. Document Type Definition (DTD)

- ◆ Defines Structure of the Document
 - Allowable tags and their attributes
 - Attribute values constraints
 - Nesting of tags
 - Number of occurrences for tags
 - Entity definitions

DTD Examples

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!ELEMENT perennials (daylily)*>
<!ELEMENT daylily (cultivar, award*, bloom, cost)+>
<!ATTLIST daylily
status (in-stock | limited | sold-out) #REQUIRED>
<!ELEMENT cultivar (#PCDATA)>
<!ELEMENT award (name, year)>
<!ELEMENT name (#PCDATA)>
<!ATTLIST name note CDATA #IMPLIED>
<!ELEMENT year (#PCDATA)>
<!ELEMENT bloom (#PCDATA)>
<!ATTLIST bloom code (E | EM | M | ML | L | E-L) #REQUIRED>
<!ELEMENT cost (#PCDATA)>
<!ATTLIST cost discount CDATA #IMPLIED>
<!ATTLIST cost currency (US | UK | CAN) "US">
```

3.2. XML Schema

- ◆ W3C recommendation released May 2001
 - – <http://www.w3.org/TR/xmlschema-0/>
 - – <http://www.w3.org/TR/xmlschema-1/>
 - – <http://www.w3.org/TR/xmlschema-2/>
 - Depends on following specifications
 - ◆ XML-Infoset, XML-Namespaces, XPath
- ◆ Benefits:
 - Standard and user-defined data types
 - Express data types as patterns
 - Higher degree of type checking
 - Better control of occurrences
- ◆ Clearly the future ... but limited support

XML Schema Example

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="perennials" type="PerennialType"/>
  <xsd:complexType name="PerennialType" >
    <xsd:element name="daylily" type="DaylilyType"
      maxOccurs="unbounded"/>
  </xsd:complexType>
  <xsd:complexType name="DaylilyType" >
    <xsd:sequence>
      <xsd:element name="cultivar" type="xsd:string"/>
      <xsd:element name="award" type="AwardType"
        maxOccurs="unbounded"/>
      <xsd:element name="bloom" type="xsd:string"/>
      <xsd:element name="cost" type="xsd:decimal"/>
    </xsd:sequence>
    <xsd:attribute name="status" type="StatusType"
      use="required"/>
  </xsd:complexType>
```

XML Schema Example (2)

```
<xsd:simpleType name="StatusType">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="in-stock"/>
    <xsd:enumeration value="limited"/>
    <xsd:enumeration value="sold-out"/>
  </xsd:restriction>
</xsd:simpleType>
...
</xsd:schema>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE university[          // --> university as root element
<!ELEMENT university (student*)> // --> university has * = Multiple students
<!ELEMENT student (name,year)>   // --> Student has elements name and year
<!ELEMENT name (#PCDATA)>       // --> name as Parsed character data
<!ELEMENT year (#PCDATA)>        // --> year as Parsed character data
]>

<university>
  <student>
    <name>
      John Niel           //----> I can also use an Integer,not good
    </name>
    <year>
      2000                //----> I can also use a string,not good
    </year>
  </student>
</university>
```

```

<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

<xsd:complexType name ="uniType">                                //--- complex datatype uniType
  <xsd:sequence>
    <xsd:element ref="student" maxOccurs="unbounded"/> //--- has unbounded no.of stu
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="stuType">                                //--- complex datatype stuType
  <xsd:sequence>
    <xsd:element ref="name"/>                            //--- has element name
    <xsd:element ref="year"/>                           //--- has element year
  </xsd:sequence>
</xsd:complexType>

<xsd:element name="university" type="uniType"/>                //--- university of type UniT
<xsd:element name="student" type="stuType"/>                  //--- student of type stuType
<xsd:element name="name" type="xsd:string"/>                   //--- name of datatype string
<xsd:element name="year" type="xsd:integer"/>                 //--- year of datatype integer
</xsd:schema>

```

```

<?xml version="1.0" encoding="UTF-8"?>
<university>
  <student>
    <name>
      John Niel
    </name>
    <year>
      2000
    </year>
  </student>
</university>

```

//--- only an Integer value is allowed

Content

1. XML and XHTML Overview
2. XML Components
3. DTD & XML Schema
4. XML Validation
5. XML Applications

4. XML Validation

◆ DTD Validation

- Process of checking a document against a DTD
 - ◆ Correct syntax
 - ◆ Correct structure
- If the document is invalid, a user agent may not be able to handle it correctly
 - ◆ parse error

Markup Validation Service

- ◆ Validator for HTML
 - URI, Local File or Direct Input
- ◆ <http://validator.w3.org>

The screenshot shows the homepage of the W3C Markup Validation Service. At the top, there is a blue header bar with the W3C logo and the text "Markup Validation Service" and "Check the markup (HTML, XHTML, ...) of Web documents". Below the header, there are three tabs: "Validate by URI" (which is selected), "Validate by File Upload", and "Validate by Direct Input". Under the "Validate by URI" tab, there is a form with a label "Address:" and a text input field. Below the input field is a link "More Options". At the bottom of the form is a "Check" button. At the very bottom of the page, there is a footer note: "This validator checks the [markup validity](#) of Web documents in HTML, XHTML, SMIL, MathML, etc. If you wish to validate specific content such as [RSS/Atom feeds](#) or [CSS stylesheets](#), [MobileOK content](#), or to [find broken links](#), there are [other validators and tools](#) available."

Validator Results

This document was successfully checked as XHTML 1.0 Strict!

Result:	Passed
Address :	<input type="text" value="http://www.w3.org/"/>
Encoding :	utf-8 <input type="button" value="(detect automatically)"/>
Doctype :	XHTML 1.0 Strict <input type="button" value="(detect automatically)"/>
Root Element:	html
Root Namespace:	http://www.w3.org/1999/xhtml



The W3C validators rely on community support for hosting and development.

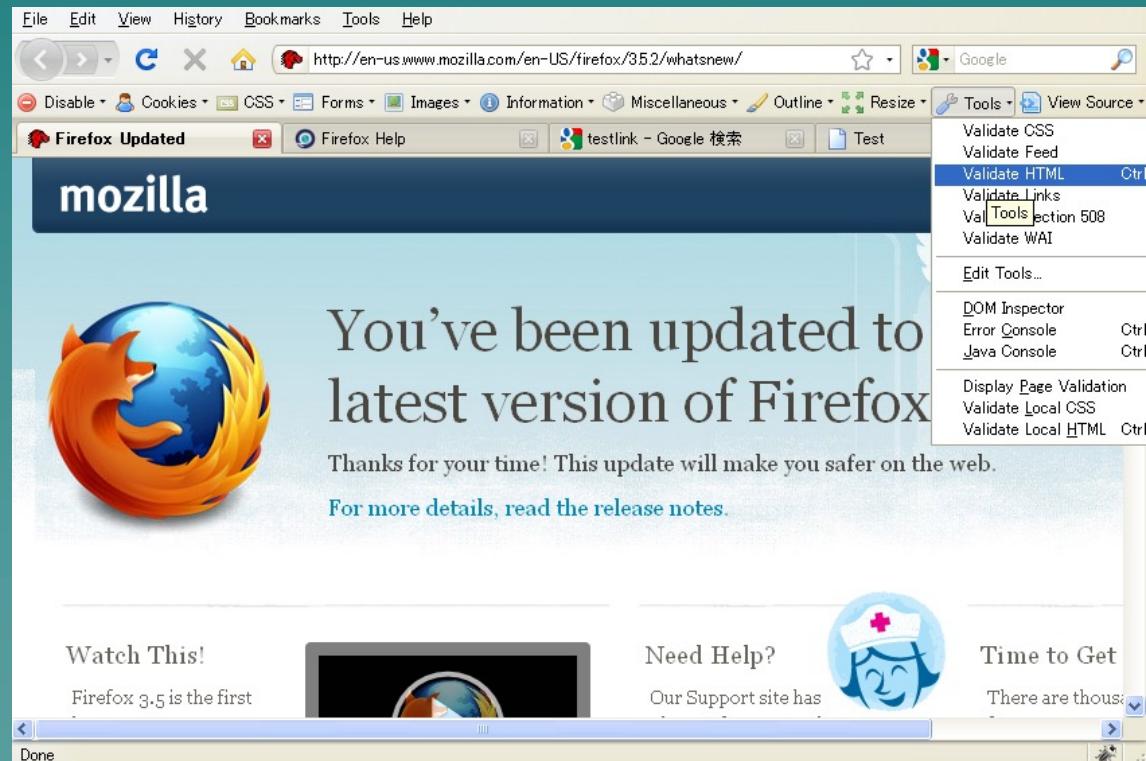
[Donate](#) and help us build better tools for a better web.

Errors found while checking this document as HTML 4.01 Transitional!

Result:	34 Errors, 8 warning(s)
Address :	<input type="text" value="http://www.yahoo.com/"/>
Encoding :	utf-8 <input type="button" value="(detect automatically)"/>
Doctype :	HTML 4.01 Transitional <input type="button" value="(detect automatically)"/>
Root Element:	html

Web Developer Tool with Validator

- ◆ A link to the validation service on the Tool menu
 - It posts the URI of the current page to the validator



Content

1. XML and XHTML Overview
2. XML Components
3. DTD & XML Schema
4. XML Validation
5. XML Applications

5. XML Application

- ◆ MathML
 - Mathematical expressions
- ◆ SVG (Scalable Vector Graphics)
 - 2D graphics applications and images
- ◆ KML (Keyhole Markup Language)
 - Geographical data for Google Earth, Maps, etc...
- ◆ XUL (XML User Interface Language, /'zu:l/)
 - GUI descriptions for Mozilla project applications (firefox)
- ◆ EPUB (Electronic PUBLications)
 - E-book description standard
- ◆ ATOM
 - Web content and metadata syndication format
 - Replacement of RSS

XML Namespace

- ◆ A way to use various XML applications as components for a document
 - Ex) HTML + MathML + SVG



Inside the `foreignObject` element, math expression like $y = \frac{1}{\sqrt{x^2+1}}$ can be put inside XHTML as well as ruby like W W W.

The base direction of the following paragraph is rtl.

WorldWideWeb
DTD
Adam

Display math inside SVG:

$$\det \begin{vmatrix} c_0 & c_1 & c_2 & \dots & c_n \\ c_1 & c_2 & c_3 & \dots & c_{n+1} \\ c_2 & c_3 & c_4 & \dots & c_{n+2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_n & c_{n+1} & c_{n+2} & \dots & c_{2n} \end{vmatrix} > 0$$

XML Namespace (2)

- ◆ Each namespace has a URI
- ◆ `xmlns` attribute
 - Default namespace for the branch

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
  <head><title>XHTML as the host language</title></head>
  <body>
    ... XHTML content ...
    <math xmlns="http://www.w3.org/1998/Math/MathML"> ...
      MathML content ...
    </math>
  ...

```

Namespace prefix

- ◆ `xmlns:??` attribute
 - Namespace for the ?? prefix

```
<math  
    xmlns="http://www.w3.org/1998/Math/MathML"  
    <xhtml:p  
        xmlns:xhtml="http://www.w3.org/1999/xhtml">  
        XHTML Paragraph</xhtml:p>  
        <svg:svg version="1.1"  
            xmlns:svg="http://www.w3.org/2000/svg">  
        </svg:svg>
```

5.1. MathML

- ◆ You can try with firefox > 3.6
 - <http://www.mozilla.org/projects/mathml/start.xhtml>

```
<mrow xmlns="&mathml;">
  <mi>x</mi><mo>=</mo>
  <mfrac>
    <mrow>
      <mrow><mo>-</mo><mi>b</mi></mrow>
      <mo>&PlusMinus;</mo>
      <msqrt><mrow>
        <msup><mi>b</mi><mn>2</mn></msup>
        <mo>-</mo>
      </mrow></msqrt>
    </mrow>
    <mrow><mn>4</mn><mi>a</mi><mi>c</mi></mrow>
    </mrow></mfrac>
  </mrow>
```

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

MathML example – Doctype and xmlns

- ◆ Both of XHTML and MathML vocabulary in the same document

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1 plus MathML
2.0//EN" http://www.w3.org/Math/DTD/mathml2/xhtml-math11-f.dtd [
<!ENTITY mathml "http://www.w3.org/1998/Math/MathML"> ]>
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">

...
<mrow xmlns="&mathml;">
```

5.2. KML (Keyhole Markup Language)

- ◆ Display geographic data in an Earth browser such as Google Earth, Google Maps,
- ◆ Example: sample.kml

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2">
  <Placemark>
    <name>HUST placemark</name>
    <description>Location of HUST</description>
    <Point>
      <coordinates>105.84413,21.00438,0</coordinates>
    </Point>
  </Placemark>
</kml>
```

To open KML files

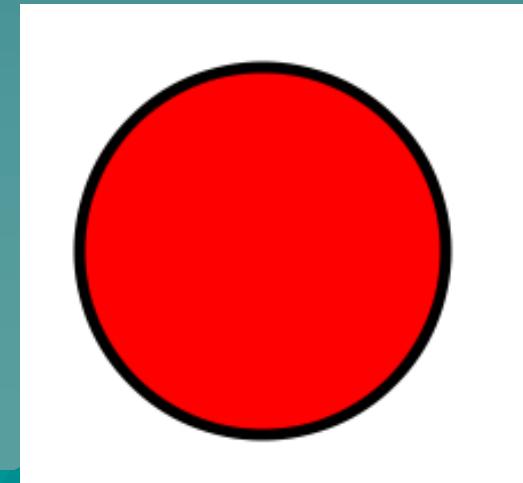
- ◆ Google Earth: Open from the file menu
- ◆ Google Map: maps.google.com
 - “My Maps” on the left sidebar
 - Use “import” menu
 - You need google account
- ◆ KML Tutorial
 - http://code.google.com/intl/en/apis/kml/documentation/kml_tut.html

5.3. SVG (Scalable Vector Graphics)

- ◆ 2D vector graphics applications and images
- ◆ You can try with firefox > 3.6
 - http://commons.wikimedia.org/wiki/SVG_examples
 - <http://www.carto.net/papers/svg/samples/>

```
<?xml version="1.0"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
 "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">

<svg xmlns="http://www.w3.org/2000/svg"
 width="200" height="200">
<circle cx="100" cy="100" r="50" stroke="black"
 stroke-width="5" fill="red" />
</svg>
```



Standalone SVG document example

◆ Doctype and `svg` element

```
<?xml version="1.0"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
 "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<svg xmlns="http://www.w3.org/2000/svg" width="200" height="200">
</svg>
```

◆ Rectangular

```
<rect x="20" y="20" width="250" height="50" fill="green"
stroke="black" stroke-width="1" />
```

◆ Circle

```
<circle cx="100" cy="100" r="50" stroke="black" stroke-
width="5" fill="red" />
```