Web Technologies

Lecture 4

XML and XHTML

XML

- E**x**tensible **M**arkup **L**anguage
- Set of **rules** for **encoding** a **document** in a format readable
  - By humans, and
  - Machines
- W3C **XML 1.0 Specification**
- **Goals**
  - Simplicity
  - Generability
  - Usability

XML

- Focuses on **documents**
- Can represent **arbitrary** data
  - Those used by web services
- Many document **formats**
  - RSS
  - Atom
  - SOAP (communication and web service protocol)
  - XHTML (similar to HTML)
  - Office Open XML
  - XMPP (communication protocol)

A little history

- 1986 – Standard Generalized Markup Language (SGML)
- 1998 – SGML is reworked into XML
- 2000 – XHTML 1.0 is released
- 2001 – XHTML 1.1 is released
- 2008 XML 1.0 standard is released
- 2015 HTML 5.0 is published as a non SGML language

Markup and content

- **Markup** text starts with
  - < and ends with >
    - Example:
    - Forms **tags**
      - Start tags
        - »
      - End tags
        - »
      - Empty-element tags
        - »
  - & and ends with ;
    - Example: & â
- Everything else is **content**

Elements and attributes

- Elements are logical document components which
  - start with a start tag and ends with an end tag, or
  - Consist of only an empty-element tag
- Attributes are name-value pairs within an element
  - Except the end element
  -

XML documents

- Declaration
- Elements
- Attributes

```
<?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 characteristics

- All opened elements need to closed
- Case sensitive
  - <Div> is different from
- No shorthand features
  - Whereas in HTML we can write

    <option selected>
  - In XHTML we must write

    <option selected="selected">
- On errors the parsing of a document stops

XML namespaces

- Element names are defined by the developer
  - This leads to confusion
  - Same <table> element can have different meanings
- Solve conflicts using **prefixes**

```
<my:table>
```

- Prefixes require the definition of a **namespace**

```
<my:table xmlns:my="http://address/to/my/namespace">
```

CDATA

- Character data section
  - Specifies a **section** of content that is **marked** to be interpreted purely as **textual data**, and not markup

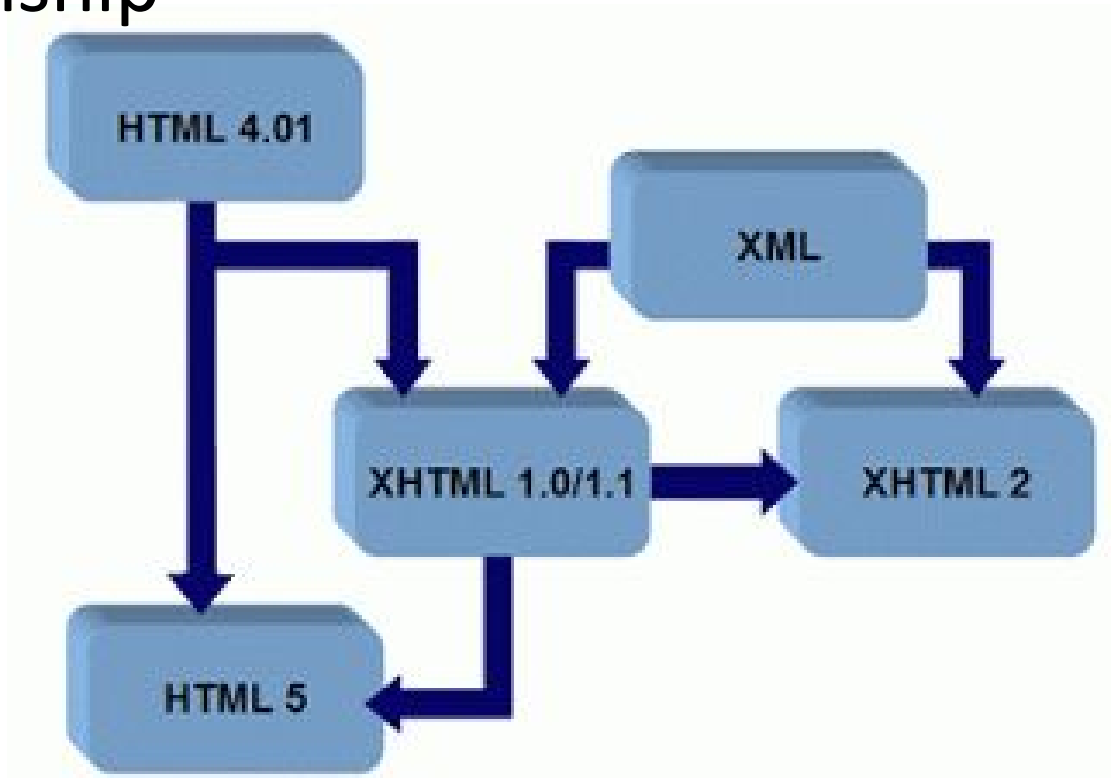
  <! [CDATA[content]]>
  - Avoids having encoded text which the parsers are not supposed to process
  - Example:
    - <![CDATA[<sender>John Smith</sender>]]>
    - Instead of **<**sender**>**John Smith**</**sender**>**

XHTML

- XHTML 1.0 is "*a reformulation of the three HTML 4 document types as applications of XML 1.0*“
- XHTML was developed to make HTML more extensible and increase interoperability with other data formats
  - Compatibility with common XML tools, servers, and proxies
  - Extensibility by adding new features such as SVG and MathML all written in XML
    - Through namespaces

XHTML and HTML

- Javascript and CSS is handled differently in XHTML
- Complex relationship



Validating XML

- **DTD** – Document Type Definition
  - Defines a standard for exchanging data
  - Example
    - Clients ensure that the data they receive from the web server is valid XHTML
  - Defined using <!DOCTYPE>
    - Example
      - <!DOCTYPE html>
- **XML** Schema
  - XML alternative to DTD

DTD vs XML schemas

- XML Schemas 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
- XML Schemas are **extensible** to additions
- XML Schemas support **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 support **namespaces**

An example

```xml
<?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>
```

DTD for the XML

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

Schema for the XML

```xml
<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>
```

Transforming XML

- **XSL** – Extensible Stylesheet Language
  - Transform and render XML documents
- **XSLT**
  - Language for transforming XML documents
- **XPath**
  - Non-XML language used to address XML elements
  - Used from inside XSLT
- Example

```
<xsl:for-each select=“./note">
```

What's next?

- Javascript
  - State vs. stateless
- Dynamic HTML manipulation
- AJAX
  - Synchronous vs. asynchronous
- JQUERY
- Server side programming
- Web services