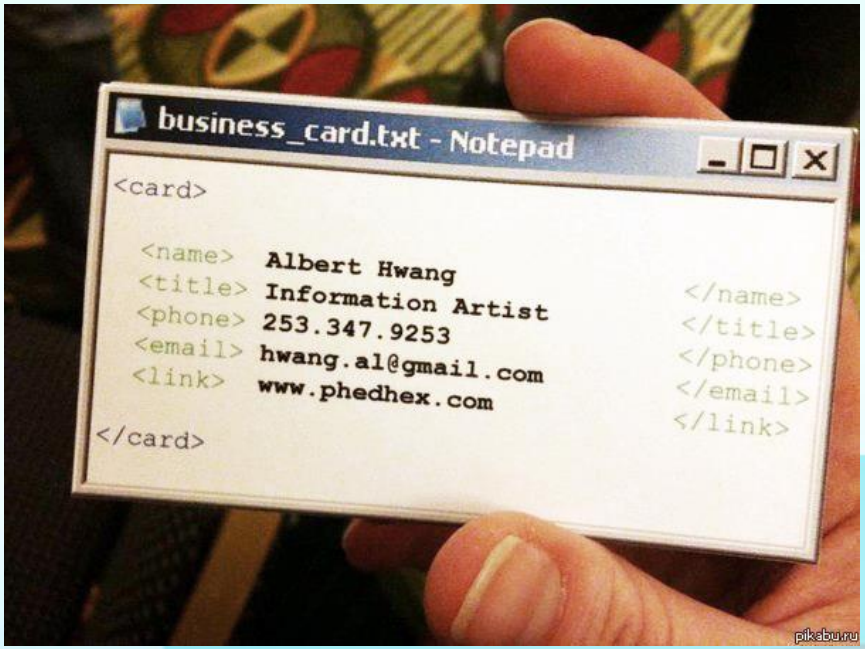
Markup Languages

Lecture 2. XML



XML

XML - eXtensible Markup Language

- XML (W3C, 1998)
- XML 1.0 (Fifth Edition) (W3C, 2008)
- XML 1.1 (Second Edition) (W3C, 2006)

Markup Languages HTML & XML

	HTML	XML
Markup Language	+	+
Tags as <name_tag></name_tag>	+	+
Document Type Definition (DTD)	+	+
Display	+	-

HTML + Web-browser = Web-page

XML + Web-browser = XML

XML and HTML were designed with different goals:

- XML was designed to transport and store data, with focus on what data is
- HTML was designed to display data, with focus on how data looks

How Can XML be Used?

- XML Separates Data from HTML
  - If you need to display dynamic data in your HTML document, it will take a lot of work to edit the HTML each time the data changes. With XML, data can be stored in separate XML files. This way you can concentrate on using HTML/CSS for display and layout, and be sure that changes in the underlying data will not require any changes to the HTML.

How Can XML be Used?

- XML Simplifies Data Sharing
  - In the real world, computer systems and databases contain data in incompatible formats. XML data is stored in plain text format. This provides a software- and hardware-independent way of storing data. This makes it much easier to create data that can be shared by different applications.

How Can XML be Used?

- XML Simplifies Data Transport
  - One of the most time-consuming challenges for developers is to exchange data between incompatible systems over the Internet. Exchanging data as XML greatly reduces this complexity, since the data can be read by different incompatible applications.

How Can XML be Used?

- XML Simplifies Platform Changes
  - Upgrading to new systems (hardware or software platforms), is always time consuming. Large amounts of data must be converted and incompatible data is often lost. XML data is stored in text format. This makes it easier to expand or upgrade to new operating systems, new applications, or new browsers, without losing data.

How Can XML be Used?

- XML is Used to Create New Internet Languages
  - A lot of new Internet languages are created with XML. Here are some examples: XHTML, WSDL for describing available web services, WAP and WML as markup languages for handheld devices, RSS languages for news feeds, RDF and OWL for describing resources and ontology, SMIL for describing multimedia for the web.

XML Syntax Rules

- The rules of construction in the XML document are described in the W3C.

XML Syntax Rules

- XML documents should begin with an **XML declaration** which specifies the version of XML being used.

```
<?xml version="1.0"?>
<greeting>Hello, world!</greeting>
```

XML Syntax Rules

- Prolog
- 1 only root element
- Hierarchy of elements
- Attributes
- Text elements
- Empty elements

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE books SYSTEM "book.dtd">
<books>
  <book id="1">
    <title>Evgeniy Onegin</title>
    <author>Alexander Pushkin</author>
    <abstract>In the book consist of
    novel in verse...</abstract>
    <shelf num="22"/>
  </book>
  <!-- ... -->
</books>
```

XML Syntax Rules

- Prolog
- 1 only root element
- Hierarchy of elements
- Attributes
- Text elements
- Empty elements

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE books SYSTEM "book.dtd">
<books>
  <book id="1">
    <title>Evgeniy Onegin</title>
    <author>Alexander Pushkin</author>
    <abstract>In the book consist of
    novel in verse...</abstract>
    <shelf num="22"/>
  </book>
  <!-- ... -->
</books>
```

XML Syntax Rules

- Prolog
- 1 only root element
- Hierarchy of elements
- Attributes
- Text elements
- Empty elements

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE books SYSTEM "book.dtd">
<books>
   <book id="1">
      <title>Evgeniy Onegin</title>
      <author>Alexander Pushkin</author>
      <abstract>In the book consist of
      novel in verse...</abstract>
      <shelf num="22"/>
   </book>
   <!-- ... -->
</books>
```

XML Syntax Rules

- Element name is case sensitive

  <aaa> ≠ <Aaa>

- All elements have to be close
- Elements may no be intersected (**<a>**)

XML Syntax Rules

- Prolog
- 1 only root element
- Hierarchy of elements
- Attributes
- Text elements
- Empty elements

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE books SYSTEM "book.dtd">
<books>
  <book id="1">
    <title>Evgeniy Onegin</title>
    <author>Alexander Pushkin</author>
    <abstract>In the book consist of
    novel in verse...</abstract>
    <shelf num="22"/>
  </book>
  <!-- ... -->
</books>
```

XML Syntax Rules

- Prolog
- 1 only root element
- Hierarchy of elements
- Attributes
- Text elements
- Empty elements

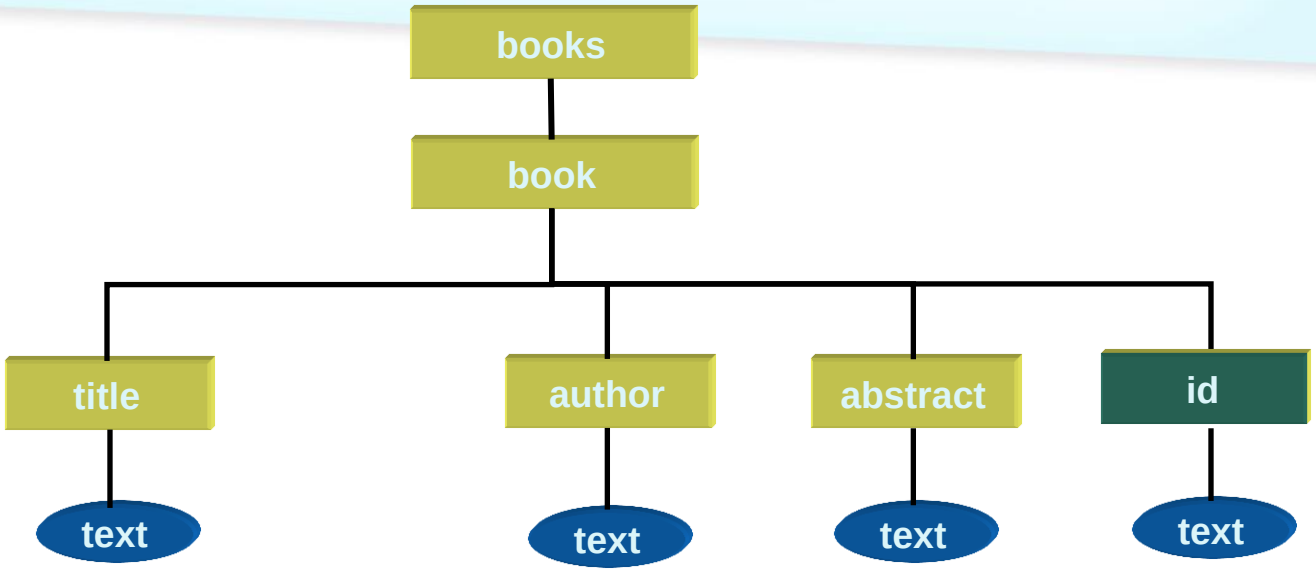
```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE books SYSTEM "book.dtd">
<books>
  <book id="1">
    <title>Evgeniy Onegin</title>
    <author>Alexander Pushkin</author>
    <abstract>In the book consist of
    novel in verse...</abstract>
    <shelf num="22"/>
  </book>
  <!-- ... -->
</books>
```

XML Syntax Rules

- Prolog
- 1 only root element
- Hierarchy of elements
- Attributes
- Text elements
- Empty elements

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE books SYSTEM "book.dtd">
<books>
  <book id="1">
    <title>Evgeniy Onegin</title>
    <author>Alexander Pushkin</author>
    <abstract>In the book consist of
    novel in verse...</abstract>
    <shelf num="22"/>
  </book>
  <!-- ... -->
</books>
```

XML Tree




```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE books SYSTEM "book.dtd">
<books>
  <book id="1">
    <title>Evgeniy Onegin</title>
    <author>Alexander Pushkin</author>
    <abstract>In the book consist of
    novel in verse...</abstract>
    <shelf num="22"/>
  </book>
  <!-- ... -->
</books>
```

Well-formed & Valid

- A "***Well-formed***" XML document has correct XML syntax.
- A "***Valid***" XML document is a "Well Formed" XML document, which also conforms to the rules of a Document Type Definition (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.

DTD for XML

- For HTML it is written DTD:
  - <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
  - <!DOCTYPE html>
  - Other
- XML have no DTD. You write DTD!

Why Use a DTD?

- With a DTD, each of your XML files can carry a description of its own format.
- With a DTD, independent groups of people can agree to use a standard DTD for interchanging data.
- Your application can use a standard DTD to verify that the data you receive from the outside world is valid.
- You can also use a DTD to verify your own data.

DTD Declaration

- If the DTD is declared **inside the XML file**, it should be wrapped in a DOCTYPE definition with the following syntax:

```
<!DOCTYPE root-element [element-declarations]>
```

- Declared in an **external file**:

```
<!DOCTYPE root-element SYSTEM "filename.dtd">
```

example.xml

```
<?xml version="1.0"?>
<!DOCTYPE book SYSTEM
"example.dtd">
<book>
…
</book>
```

example.dtd

```
<?xml version="1.0"?>
<!ELEMENT book (title, author*,
pages)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT author (#PCDATA)>
<!ELEMENT pages (#PCDATA)>
```

The basic markup declarations

- Elements
- Attributes
- Entity
- PCDATA
- CDATA

Elements

- Elements are the main building blocks of both XML and HTML documents.

```
<!ELEMENT name (element-content)>
```

- element-content:
  - #PCDATA
  - child-element-name, child-element-name, …
  - EMPTY
  - ANY

Element Example

- example.dtd:

```
<!ELEMENT books (#PCDATA)>
```

- Valid:
  - ```
    <!DOCTYPE  books SYSTEM "example.dtd">
    <books>Any text</books>
    ```
  - ```
    <!DOCTYPE books SYSTEM "example.dtd">
    <books/>
    ```
- No valid:
  - ```
    <!DOCTYPE books SYSTEM "example.dtd">
    <text>Any text</text>
    ```
    ```

Element Example

- example.dtd:

```
<!ELEMENT book (title, author)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT author (#PCDATA)>
```

- Valid:
  -
```
<book>
   <title>Evgeniy Onegin</title>
   <author>Alexander Pushkin</author>
</book>
```
- No valid:
  -
```
<book>
   <title>Evgeniy Onegin</title>
</book>
```
  -
```
<book>
   <author>Alexander Pushkin</author>
   <title>Evgeniy Onegin</title>
</book>
```

Error: no element author!

Error : incorrect order of the elements!

Element Example

- example.dtd:

```
<!ELEMENT shelf (EMPTY)>
```

- Valid:
  - `<shelf />`
  - `<shelf></shelf>`
- No valid:
  - `<shelf>123</shelf>`

Error: Element shelf empty no!

Number of occurrences of elements in the document

- Declaring Only One Occurrence of an Element

```
<!ELEMENT name (child-element-name)>
```

- Declaring Minimum One Occurrence of an Element

```
<!ELEMENT name (child-element-name+)>
```

- Declaring Zero or More Occurrences of an Element

```
<!ELEMENT name (child-element-name*)>
```

- Declaring Zero or One Occurrences of an Element

```
<!ELEMENT name (child-element-name?)>
```

or

```
<!ELEMENT name (book, (autor|autors))>
```

Example

- example.dtd:

```
<!ELEMENT book (title, author)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT author (#PCDATA)>
```

- No valid XML-документы:
  -

```
<book>
    <title>Evgeniy Onegin</title>
    <title>Evgeniy Onegin</title>
    <author>Alexander Pushkin</author>
</book>
```

Error: element title must be one!

Example

- example.dtd:

```
<!ELEMENT books (book+)>
```

- Valid:
  -
```
<books>
    <book>...</book>
    <book>...</book>
</books>
```
- No valid:
  -
```
<books></books>
```

Error: A required element book!

Example

- example.dtd:

```
<!ELEMENT books (book*)>
```

- Valid:
  - <books>
    <book>...</book>
    <book>...</book>
    </books>
- ~~No~~ valid:
  - <books></books>

Example

- example.dtd:

```
<!ELEMENT books (book?)>
```

- Valid:
  -
```
<books>
    <book>...</book>
</books>
```
  -
```
<books></books>
```
- No valid:
  -
```
<books>
    <book>...</book>
    <book>...</book>
</books>
```

Error: element book must be not more than one!

Declaring Mixed Content

```
<!ELEMENT book (#PCDATA|title|author)*>
```

- The example declares that the "book" element can contain zero or more occurrences of parsed character data (**#PCDATA**), "**title**" or "**author**" elements.

Attributes

- Attributes provide **extra information about elements**.

```
<!ATTLIST element-name attr-name attr-type
default-value>
```

- attr-type:
  - Type: CDATA, ID, IDREF, NMTOKEN,…
  - Values: (en1|en2|…)
- default-value:
  - value
  - #REQUIRED
  - #IMPLIED
  - #FIXED value

Example

- example.dtd:

```
<!ELEMENT shelf (EMPTY)>
<!ATTLIST shelf num CDATA #REQUIRED>
```

- Valid:
  - `<shelf num="123"/>`
  - `<shelf num="s1"/>`
- No valid:
  - `<shelf/>`

Error: A required attribute num!

Example

- example.dtd:

```
<!ELEMENT book (ANY)>
<!ATTLIST book id ID #REQUIRED>
```

- Valid:
  -
```
<books>
  <book id="a1">Text</book>
  <book id="a2">Text</book>
</books>
```
- No valid:
  -
```
<books>
  <book id="a1">Text</book>
  <book id="a1">Text</book>
</books>
```
  -
```
<books>
  <book id="1">Text</book>
</books>
```

Error: Attrinute id must be unique!

Error: The value of an ID attribute type must begin with a letter!

Example

- example.dtd:

```
<!ELEMENT shelf (EMPTY)>
<!ATTLIST shelf
num (first|second) #REQUIRED>
```

- Valid:
  - `<shelf num="first"/>`
  - `<shelf num="second"/>`
- No valid:
  - `<shelf num="third"/>`

Error: Attribute value is incorrect!

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: " ". 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.
- Example:
  - & = &
  - < = <
  - > = >

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.**
- 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 & < and > 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.