Extensible Markup Language (XML)

CSE 190 M (Web Programming), Spring 2007
University of Washington

Reading: Sebesta Ch. 8 sections 8.1 - 8.3, 8.7 - 8.8, 8.10.3

What is XML?

- a specification for creating markup languages to store hierarchical data
- used to facilitate sharing data between different systems
- XHTML is a subset of XML
  - a language created using XML specifications
  - an adaptation of old HTML to fit XML's syntax requirements

Structure of an XML document

- a header, then a single document tag that can contain other tags

```
<?xml version="1.0" encoding="UTF-8"?>
document tag
```

- tag syntax:

```
<element attributes>
    text or tags
</element>
```

  - or a tag with no inner tags/content can end with />
- attribute syntax:

```
name="value"
```

- comments: `<!-- comment -->`

An example XML file

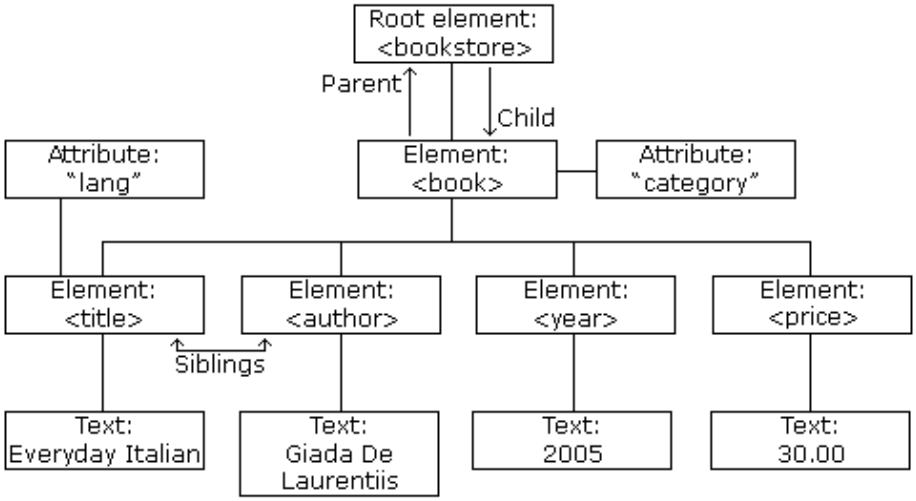
```
<?xml version="1.0" encoding="UTF-8"?>
<note>
    <to>Tove</to>
    <from>Jani</from>
    <subject>Reminder</subject>
    <message>
        Don't forget me this weekend!
    </message>
</note>
```

- other examples: music, math, vector graphics, web feeds

Larger XML file example

```
<?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="computers">
  <title lang="en">XQuery Kick Start</title>
  <author>James McGovern</author>
  <year>2003</year><price>49.99</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="computers">
  <title lang="en">Learning XML</title>
  <author>Erik T. Ray</author>
  <year>2003</year><price>39.95</price>
</book></bookstore>
```

Resulting tree structure (partial)



- the XML tags have a tree structure
- nodes have parents, children, and siblings
- (we'll process these nodes using the Javascript XML DOM)

What tags are legal in XML?

- *any tag you want*; you can make up your own structure
- schema: description of which tags are legal with your data
- a schema can be formally defined in different languages:
  - Document Type Definition (DTD)
  - W3C XML Schema
- if you define a formal schema for your XML files, you can use tools to validate XML files to make sure they match your schema
  - XHTML is an XML language that has a schema
  - allows W3C validator to check HTML files to see if they follow it

Facts about XML data

- comes from many sources on the web:
  - web servers store data as XML files
  - databases sometimes return query results as XML
  - web services use XML to communicate
  - RSS news feeds use an XML format
- pros and cons of XML:
  - pro:
    - human-readable, self-documenting format
    - strict syntax allows standardized tools
    - international, platform-independent
    - can represent almost any general kind of data (record, list, tree)
  - con:
    - bulky syntax/structure makes files large; can decrease performance

Displaying XML data on a web page

- XML is pure data; doesn't specify how it should be displayed
- can transform XML into HTML using Javascript XML DOM
- basic outline:
  - fetch XML data using an Ajax `XMLHttpRequest` object
  - retrieve the resulting data as an XML document tree
  - examine the tree, using DOM properties we've already seen
  - turn XML data into HTML tags as desired
- other ways to transform XML (not covered): CSS, XSLT

Ajax `XMLHttpRequest` template for XML

```
var ajax = new XMLHttpRequest();
ajax.onreadystatechange = function() {
    if (ajax.readyState == 4) {
        do something with ajax.responseXML;
    }
};
ajax.open("GET", url, true);
ajax.send(null);
```

- `responseXML` returns response as an XML document tree
- can use all DOM properties and methods on `responseXML` tree

DOM node properties/methods

- properties:
  - `firstChild`, `lastChild`, `childNodes`, `nextSibling`, `previousSibling`, `parentNode`
  - **`nodeName`**, **`nodeType`**, **`nodeValue`**, **`attributes`**
- methods:
  - `appendChild`, `insertBefore`, `removeChild`, `replaceChild`
  - **`getElementsByTagName`**, **`getAttribute`**, **`hasAttributes`**, **`hasChildNodes`**

Details about XML node properties

- `nodeType` : what kind of node it is

Kind of node	**`nodeType` value**
element	1
attribute	2
text	3
comment	8
document	9

- `nodeName` : uppercase version of tag such as `"DIV"` or `"ARTICLE"`
  - an attribute node's name is the attribute's name
  - all text nodes have name `"#text"`
  - document node has name `"#document"`
- `nodeValue` : text inside a text node, or value of an attribute node

Navigating the node tree

```
element.getElementsByTagName("tag")
```

- get an array of all children of the given type (`"p"`, `"div"`, etc.)
- can be called on the overall `document` or on a specific node

```
element.getAttribute("attributeName")
```

- get an attribute from an element node

Recall: XML file

```
<?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="computers">
  <title lang="en">XQuery Kick Start</title>
  <author>James McGovern</author>
  <year>2003</year><price>49.99</price>
</book>
...
</bookstore>
```

- let's write code to process the computer books in this file

Navigating node tree example

```
var xmlDoc = ajax.responseXML;
var books = xmlDoc.getElementsByTagName("book");
for (var i = 0; i < books.length; i++) {
    var category = books[i].getAttribute("category");
    if (category == "computers") {
        var title = books[i].getElementsByTagName("title")[0].firstChild.nodeValue;
        var author = books[i].getElementsByTagName("author")[0].firstChild.nodeValue;

        var p = document.createElement("p");
        p.innerHTML = title + ", by " + author;
        document.body.appendChild(p);
    }
}
```

- makes a paragraph for each `<book>` element about computers
  (e.g. "XQuery Kick Start, by James McGovern")

Practice problem: Animal game

- Write a program where the user thinks of an animal, and the page repeatedly asks yes/no questions to find it.
- The data comes from a web application (`game.php`) that serves XML data in the following format:

```
<node id="id">
    <question>question</question>
    <yes id="id" />
    <no id="id" />
</node>

<node id="id">
    <answer>answer</answer>
</node>
```

- to get a node with a given id: `grades.php?id=id`