中山大學
SUN YAT-SEN UNIVERSITY

Lecture 20
Pragmatic XML

SE-805 Web 2.0 Programming (supported by Google)

http://my.ss.sysu.edu.cn/courses/web2.0/

School of Software, Sun Yat-sen University

Outline

- **XML Basic**
- XML and Ajax
- Programming with XML

What is XML?

- **XML(eXtensible Markup Language)**: a "skeleton" for creating markup languages
- You already know it!
  - Syntax is identical to XHTML's:

```
<element attribute="value">content</element>
```
XML

- Languages written in XML specify:
  - Names of tags in XHTML: `h1`, `div`, `img`, etc.
  - Names of attributes in XHTML: `id`/`class`, `src`, `href`, etc.
  - Rules about how they go together
    in XHTML: `inline` vs. `block`-level elements
- Used to present complex data in human-readable form
  - "Self-describing data"

Anatomy of an XML File

```xml
<?xml version="1.0" encoding="UTF-8"?>   <!-- XML prolog -->
<note>   <!-- root element -->
  <to>Tove</to>
  <from>Jani</from>   <!-- element ("tag") -->
  <subject>Reminder</subject>   <!-- content of element -->
  <message language="english">   <!-- attribute and its value -->
    Don't forget me this weekend!
  </message>
</note>
```

XML

- Begins with an <?xml ... ?> header tag ("**prolog**")
- Has a single **root** element (in this case, **note**)
- Tag, attribute, and comment syntax is just like XHTML

Use of XML

- 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
- XML is the *de facto* universal format for exchange of data
- XML languages are used for music, math, vector graphics
- Popular use: RSS for news feeds & podcasts

Pros and Cons of XML

- **Pros**:
  - Easy to read (for humans and computers)
  - Standard format makes automation easy
  - Don't have to "reinvent the wheel" for storing new types of data
  - International, platform-independent, open/free standard
  - Can represent almost any general kind of data (record, list, tree)
- **Cons**:
  - Bulky syntax/structure makes files large; can decrease performance
    - Example: quadratic formula in MathML
  - Can be hard to "shoehorn" data into a good XML format

What Tags are Legal in XML?

- *Any tags you want!*
- Examples:
  - An email message might use tags called to, from, subject
  - A library might use tags called book, title, author
- When designing an XML file, ***you*** choose the tags and attributes that best represent the data
- Rule of thumb: data = tag, metadata = attribute

Doctypes and Schemas

- "**Rule books**" for individual flavors of XML
  - list which tags and attributes are valid in that language, and how they can be used together
- Used to *validate* XML files to make sure they follow the rules of that "flavor"
  - The W3C HTML validator uses the XHTML doctype to validate your HTML
- For more info:
  - Document Type Definition (DTD) ("doctype")
  - W3C XML Schema
- Optional — if you don't have one, there are no rules beyond having well-formed XML syntax

Outline

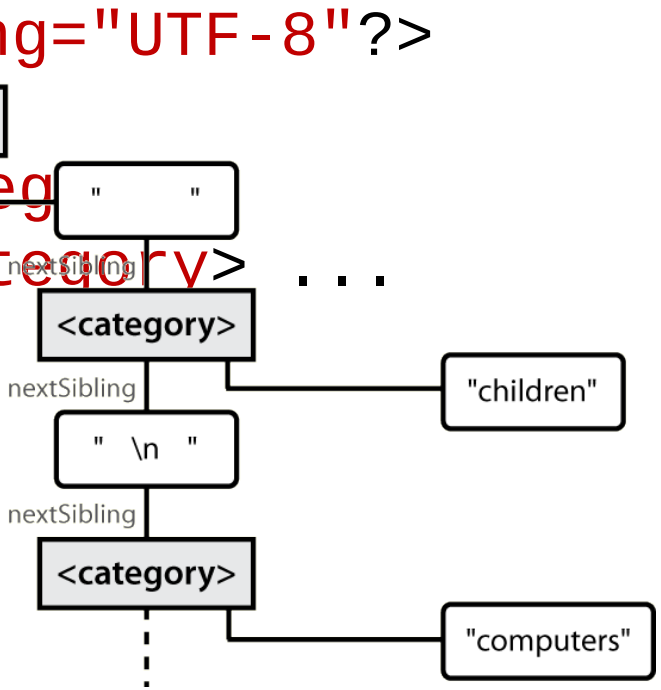
- XML Basic
- **XML and Ajax**
- Programming with XML

XML and AJAX

- Web browsers can display XML files, but often you instead want to fetch one and analyze its data
- The XML data is fetched, processed, and displayed using AJAX
  - (XML is the "X" in "AJAX")
- It would be very clunky to examine a complex XML structure as just a giant string!
- Luckily, the browser can break apart (**parse**) XML data into a set of objects
  - There is an XML **DOM**, very similar to the (X)HTML DOM

XML DOM Tree Structure

```
<?xml vers encoding="UTF-8"?>
<categories>
    <category>children</categ
    <category>computers</category> ...
</categories>
```



- The XML tags have a tree structure
- DOM nodes have parents, children, and siblings

Recall: Javascript XML (XHTML) DOM

- The **DOM** properties and methods* we already know can be used on XML nodes:
- Properties:
  - firstChild, lastChild, childNodes, nextSibling, previousSibling, parentNode
  - **nodeName**, **nodeType**, **nodeValue**, **attributes**
- Methods:
  - appendChild, insertBefore, removeChild, replaceChild
  - **getElementsByTagName**, **getAttribute**, **hasAttributes**, **hasChildNodes**
- Caution: cannot use HTML-specific properties like innerHTML in the XML DOM!

* (Though not Prototype's, such as up, down, ancestors, childElements, descendants, or siblings)

Navigating the Node Tree

- **Caution**: can ***only*** use standard DOM methods and properties in XML DOM HTML DOM has Prototype methods, but XML DOM **does not!**
- **Caution**: can't use ids or classes to use to get specific nodes
  - id and class are not necessarily defined as attributes in the flavor of XML being read
- **Caution**: firstChild/nextSibling properties are unreliable
  - annoying whitespace text nodes!
- The best way to walk the XML tree:

```
var elms = node.getElementsByTagName("tagName")
```
JS

  - Returns an **array** of all *node*'s children of the given tag name

```
node.getAttribute("attributeName")
```
JS

  - Gets an attribute of an element

Outline

- XML Basic
- XML and Ajax
- **Programming with XML**

Using XML Data in a Web Page

- Procedure:

1. Use Ajax to fetch data
2. Use DOM methods to examine XML:
   - *XMLnode*.getElementsByTagName()
3. Extract the data we need from the XML:
   - *XMLelement*.getAttribute(), *XMLelement*.firstChild.nodeValue, etc.
4. Create new HTML nodes and populate with extracted data:
   - document.createElement(), *HTMLelement*.innerHTML
5. Inject newly-created HTML nodes into page
   - *HTMLelement*.appendChild()

Fetching XML using AJAX (template)

```
  new Ajax.Request(
    "url",
    {
      method: "get",
      onSuccess: functionName
    }
  );
  ...

function functionName(ajax) {
  do something with ajax.responseXML;
}
```
JS

- `ajax.responseText` contains the XML data in plain text
- `ajax.responseXML` is a pre-parsed XML DOM object

Analyzing a Fetched XML File Using DOM

```xml
<?xml version="1.0" encoding="UTF-8"?>
<foo bloop="bleep">
  <bar/>
  <baz><quux/></baz>
  <baz><xyzzy/></baz>
</foo>
```
XML

- We can use **DOM** properties and methods on `ajax.responseXML`:

```js
// zeroth element of array of length 1
var foo = ajax.responseXML.getElementsByTagName("foo")[0];

// ditto
var bar = foo.getElementsByTagName("bar")[0];

// array of length 2
var all_bazzes = foo.getElementsByTagName("baz");

// string "bleep"
var bloop = foo.getAttribute("bloop");
```
JS

Recall: Pitfalls of the DOM

```xml
<?xml version="1.0" encoding="UTF-8"?>
<foo bloop="bleep">
  <bar/>
  <baz><quux/></baz>
  <baz><xyzzy/></baz>
</foo>
```
XML

- We are reminded of some pitfalls of the **DOM**:

```js
// works - XML prolog is removed from document tree
var foo = ajax.responseXML.firstChild;

// WRONG - just a text node with whitespace!
var bar = foo.firstChild;

// works
var first_baz = foo.getElementsByTagName("baz")[0];

// WRONG - just a text node with whitespace!
var second_baz = first_baz.nextSibling;

// works - why?
var xyzzy = second_baz.firstChild;
```
JS

Larger XML File Example

```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="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>
```

XML

Navigating Node Tree Example

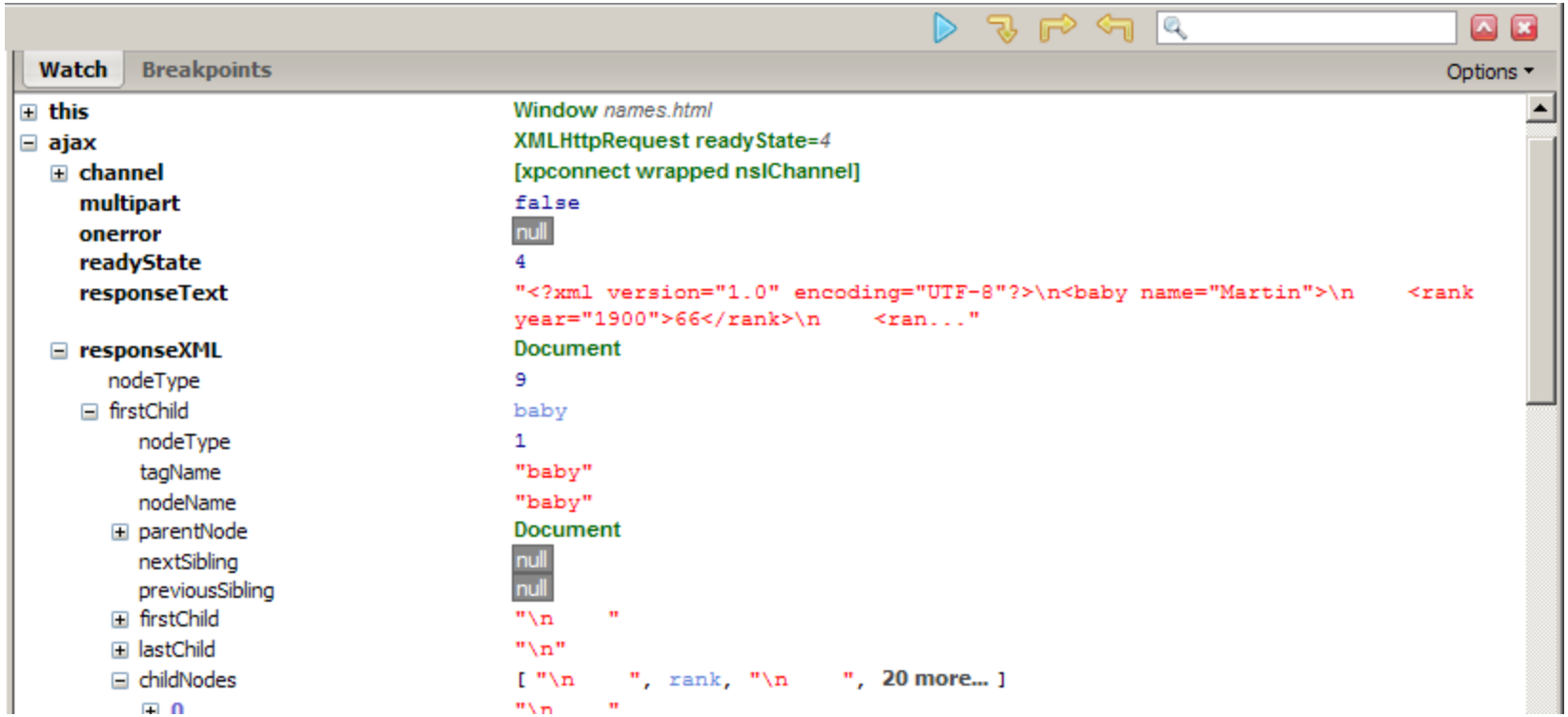
```JS
// make a paragraph for each book about computers
var books = ajax.responseXML.getElementsByTagName("book");
for (var i = 0; i < books.length; i++) {
  var category = books[i].getAttribute("category");
  if (category == "computers") {
    // extract data from XML
    var title = books[i].getElementsByTagName("title")[0].firstChild.nodeValue;
    var author = books[i].getElementsByTagName("author")[0].firstChild.nodeValue;

    // make an XHTML <p> tag containing data from XML
    var p = document.createElement("p");
    p.innerHTML = title + ", by " + author;
    document.body.appendChild(p);
  }
}
```

Historical Interlude: Why XHTML?

- In XML, different "flavors" can be combined in single document
- Theoretical benefit of including other XML data in XHTML
  - Nobody does this
- Most embedded data are in non-XML formats (e.g., Flash)
  - Non-XML data must be embedded another way (we'll talk about this later on)
- Requires browser/plugin support for other "flavor" of XML
  - Development slow to nonexistent
  - Most XML flavors are specialized uses

Debugging responseXML in Firebug



- Can examine the entire XML document, its node/tree structure

Summary

- XML Basic
  - XML is a language for specifying structured data
  - Pros and cons
  - Doctypes and Schemas
- XML and Ajax
  - XML and Ajax
  - XML DOM
  - Using XML in a web page
  - Fetching XML via Ajax
- Programming with XML
  - Manipulating and Debugging XML

Exercises

- Write a simple **Ajax** to-do list application as a Web page.
  - A element wraps all html elements
  - A form for adding new items and a list of all items
  - Buttons of "select all", "deselect all", "remove"
  - When the "add" button is clicked the new to-do item will be inserted to the bottom of the list in an Ajax way
    - Create a php script generating xml data for a new to-do item, something likes: (Content-type: application/xml)

```
<todo>
   <content>YOUR_TODO</content>
</todo>
```

    - Alter the onSuccess handler to consume the responseXML and insert the new todo into the list

Further Readings

- W3C XML Specification http://www.w3.org/XML/
- W3Schools XML tutorial http://www.w3schools.com/xml/default.asp
- W3Schools XML examples http://www.w3schools.com/XML/xml_examples.asp
- Ajax/JavaScript XML processing example/tutorial http://www.captain.at/howto-ajax-process-xml.php
- Developer Notes for prototype.js http://www.sergiopereira.com/articles/prototype.js.html

Thank you!