USER XML

XML

Announcements

- Second Test: Wednesday April 23
- Project Presentations
  - Monday, April 28
  - Wednesday, April 30

Scrum Masters

Backslash	MICHAEL	HOLUPKA
C.O.D.E.	MINGJIAN	ZHANG
Cellar	EVAN	BASTA
ContraWeb	RUBY	REYNOSO
Hacklemore	EMILY	ANSLEY
Lanister	EDWARD	SAMUALS
Llama	CHRISTOPHER	BELL
Sk3m Team	MATTHEW	NING
SqlThePrql	JEREMY	WARNER
Synapps	CHARLES	KELMAN
Tautology	TAIT	MADSEN
Team RNG	CHI MAN	WONG

Fetching XML using AJAX (template)


```
new Ajax.Request(
"url",
{
        method: "get",
        onSuccess: functionName
}
);
...
function functionName(ajax) {
        do something with ajax.responseXML;
}
```

JS

- ajax.response**Text** contains the XML data in plain text
- ajax.response**XML** 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

Standup

Discuss questions with your Scrum Team

Quiz

Quiz

- To be done in team
- Put the team name and all the members who worked on it at the top of the page

```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="computers">
                <title lang="en">Learning XML</title>
                <author>Erik T. Ray</author>
                <year>2003</year><price>39.95</price>
        </book>
</bookstore>
```
XML

Write two JavaScript functions:

1. A function that will make an Ajax request to fubar.bookstore.com
2. The function specified in your Ajax request, that will create a page consisting of the books in the “cooking” category listing the author and the title of the book.

And the answer is …

```
new Ajax.Request(”fubar.bookstore.com”, {
       method: "get",
       onSuccess: processBooks
} );

function  processBooks(ajax) {
 var books = ajax.responseXML.getElementsByTagName("book");
 var list = document.createElement(“ul”);
 for (var i = 0; i < books.length; i++) {
  var category = books[i].getAttribute(”category");
  if (category == "cooking") {
   var title = books[i].getElementsByTagName("title”[0].
                                firstChild.nodeValue;
   var author = books[i].getElementsByTagName("author”[0].
                                firstChild.nodeValue;
   var li = document.createElement(“li");
       li.innerHTML = title + ”, " + author;
       list.appendChild(ul);
  }
 }
 document.body.appendChild(list);
}
```

JS

12

XML

What is XML?

- XML: a "skeleton" for creating markup languages
- you already know it!
  - syntax is identical to XHTML's:

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

- 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

Why do we need XML?

- 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

Uses 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

pro:

- 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)

Pros and cons of XML

con:

- 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

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

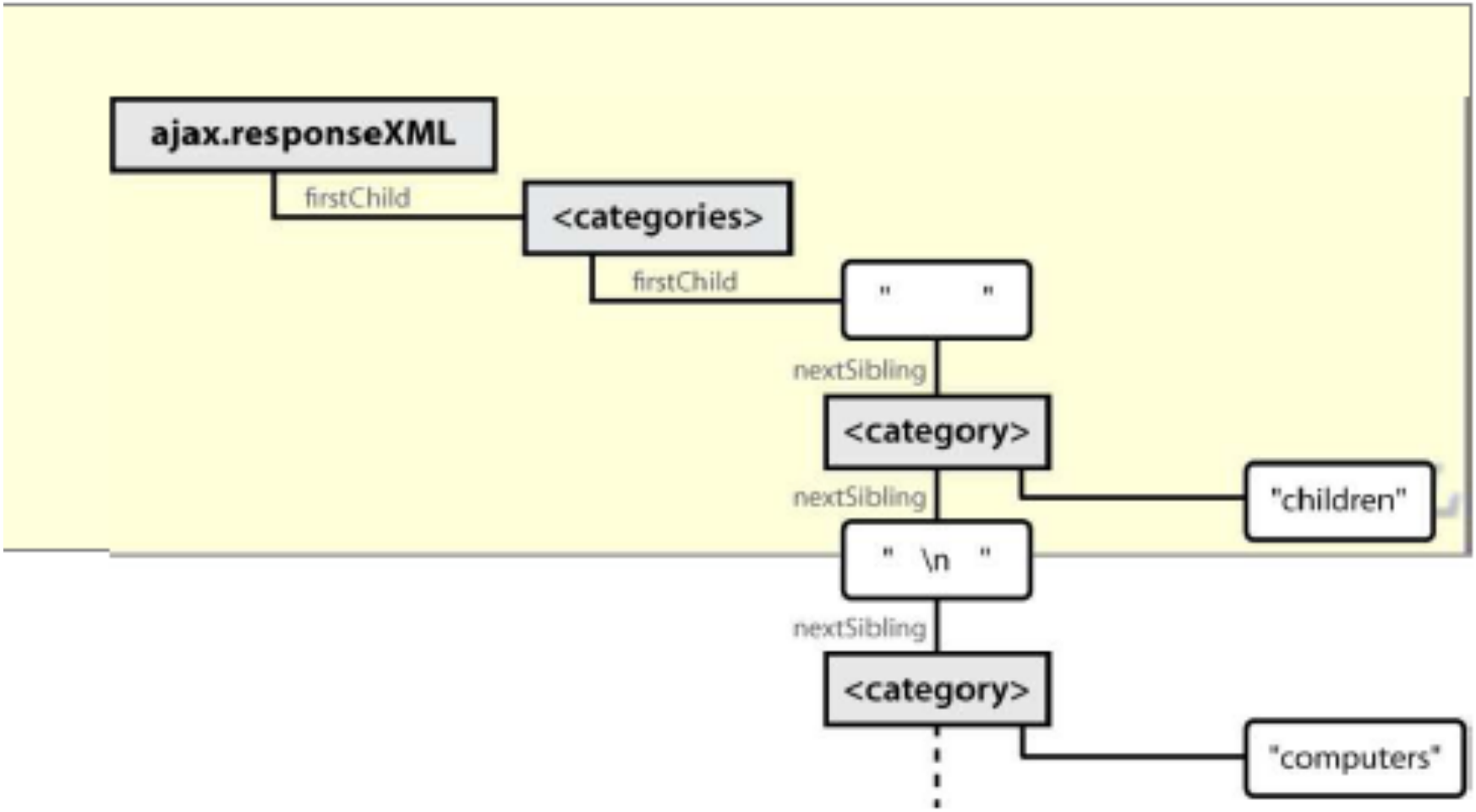
XML DOM tree structure


```xml
<?xml version="1.0" encoding="UTF-8"?>
      <categories>
            <category>children</category>
            <category>computers</category>
            ...
      </categories>
```

XML

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

XML DOM tree structure



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!

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

Navigating the node tree

- caution: firstChild/nextSibling properties are unreliable
  - annoying whitespace text nodes!
- the best way to walk the XML tree:

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

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

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

  - gets an attribute of an element

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.response**Text** contains the XML data in plain text
- ajax.response**XML** 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

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>
```

Navigating node tree example


```
// 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);
    }
}
```

JS

Resources

- http://www.sitepoint.com/really-good-introduction-xml/
- http://www.w3.org/XML/Schema.html