JavaScript: A Crash Course
Part IV: Parsing XML

Originals of Slides and Source Code for Examples:
http://courses.coreservlets.com/Course-Materials/ajax.html

Customized Java EE Training: http://courses.coreservlets.com/
Servlets, JSP, JSF 2.0, Struts, Ajax, GWT 2.0, Spring, Hibernate, SOAP & RESTful Web Services, Java 6.
Developed and taught by well-known author and developer. At public venues or onsite at your location.

For live Ajax & GWT training, see training courses at http://courses.coreservlets.com/.

Taught by the author of Core Servlets and JSP, More Servlets and JSP, and this tutorial. Available at public venues, or customized versions can be held on-site at your organization.

- Courses developed and taught by Marty Hall
  - Java 6, servlets/JSP (intermediate and advanced), Struts, JSF 1.x, JSF 2.0, Ajax, GWT 2.0 (with GXT), custom mix of topics
  - Ajax courses can concentrate on 1 library (jQuery, Prototype/Scriptaculous, Ext-JS, Dojo, Google Closure) or survey several
- Courses developed and taught by coreservlets.com experts (edited by Marty)
  - Spring, Hibernate/JPA, EJB3, Web Services, Ruby/Rails
Contact hall@coreservlets.com for details
Topics in This Section

• Motivation
• Getting document
  – Via Ajax
  – The current HTML page as DOM
  – From string (for interactive testing)
• Main XML-related classes
  – Document
  – Element
  – Node
Why Know XML Techniques in JavaScript?

• Primary reason
  – To extract information from XML sent by server in response to an Ajax request.
    • We will discuss pros and cons of sending HTML, JSON, and XML in a later lecture

• Secondary reason
  – Browser represents page internally as DOM (even if the page is written in HTML 4 or otherwise violates XML syntax).
  – So, XML-related methods can also be used to extract information from the current page

• Note
  – If you know DOM API from Java or another language, the JavaScript API is very similar

API Overview

• API is very similar to Java DOM API
  – If you know Java org.w3c.dom classes and methods, they are almost the same in JavaScript

• Document class
  – Represents top-level document
    • Also a specialized version representing the HTML page

• Element class
  – Represents XML/HTML element
  – Inherits Node methods plus has some extras

• Node class
  – Represents node in XML tree
    • Element is main node type, but there are also text nodes, CDATA notes, and a few others
  – Most Element methods inherited from here
Most Important Topics

• Very important methods and variables
  – How to treat the Ajax response as XML
    • var xmlDoc = response.responseXML;
      – Some browsers let you take response.responseText and then treat it as an XML document, but this is not portable.
  – How to get an array of subelements
    • xmlDoc.getElementsByTagName(...)
  – How to get an attribute of an element
    • someElement.getAttribute(…)
  – How to get the body content of an element
    • someElement.firstChild.nodeValue

You can do many Ajax applications (if they use XML at all!) using only these four techniques!

Second-Most Important Topics

• Medium important topics
  – The top-level element
    • xmlDoc.documentElement (and …nodeName)
  – An element’s main tag name
    • someElement.nodeName
  – All attributes of an element
    • someElement.attributes
  – Child nodes of an element
    • someElement.childNodes

This section cover a lot of other topics, and unless you have XML/DOM experience already, you won’t follow all of it. But, for most Ajax applications, these two slides are by far the most important topics to know.
Summary of Main Classes

- **Document class**
  - Properties
    - documentElement
  - Methods
    - getElementsByTagName, getElementById (HTML only)

- **Element class**
  - Methods
    - getAttribute, getElementsByTagName, hasAttribute

- **Node**
  - Properties
    - attributes, childNodes, firstChild, lastChild, nextSibling, nodeName, nodeType, nodeValue, parentNode, previousSibling
  - Methods
    - hasAttributes, hasChildNodes, normalize
Getting Document: Real Life

- **Getting general XML document**
  - You get XML document as the result of an Ajax request
  - `var xmlDocument = request.responseXML;`
- **Getting XML doc representing HTML page**
  - Use builtin “document” variable
  - Many special features apply to this specialized version
    - `getElementById` method, `innerHTML` property for elements, forms and anchors and images properties, case-insensitivity for `getElementsByTagName`, lots more specific to HTML
    - These are not general to XML, and do not apply to an XML doc that comes in over the network in response to Ajax request
  - `http://www.w3schools.com/htmldom/dom_obj_document.asp`

Getting Document: Practice and Learning

- **Firefox**
  ```javascript
  var xmlString = "<blah>...</blah>";
  var parser = new DOMParser();
  var xmlDocument = parser.parseFromString(xmlString, "application/xml");
  ```
- **Internet Explorer**
  ```javascript
  var xmlDocument =
    new ActiveXObject("Microsoft.XMLDOM");
  xmlDocument.async = false;
  xmlDocument.loadXML(xmlString);
  return (xmlDocument);
  ```
- **Figuring out which is which**
  - Check if (typeof DOMParser != "undefined")
    - True: use Firefox approach
    - False: use IE approach
  - Warning: just for practice
The Document Class

- **documentElement property**
  - The root Element of the document

- **getElementById method**
  - Returns the Element with the specified ID.
    - For HTML documents only!
      - Refers to attribute that the DTD defines as an "id attribute", not necessarily named "id". Does not match attributes named "id" in regular XML docs.
    - Is supposed to be case-sensitive, but IE is case-insensitive

- **getElementsByTagName**
  - Returns an array of Elements that have that tag name
    - Can use "*" for all Elements in document
      - Unsupported in IE 5
    - Is case-sensitive for regular XML documents
    - Is case-insensitive for HTML documents
      - Even when using XHTML

The Document Class: Examples (Code)

```javascript
function getXmlDoc(xmlString) {
  var parser = new DOMParser();
  var xmlDocument = 
    parser.parseFromString(xmlString, "application/xml");
  return(xmlDocument);
}
var test = 
  "<customers rating='vip'>" + 
  "<customer id='a1234'>" + 
  "<firstName>Rafael</firstName>" + 
  "<lastName>Nadal</lastName>" + 
  "</customer>" + 
  "<customer id='a1235'>" + 
  "<firstName>Roger</firstName>" + 
  "<lastName>Federer</lastName>" + 
  "</customer>" + 
  "</customers>";
var testDoc = getXmlDoc(test);
```
The Document Class: Examples (Results)

```python
>>> testDoc.documentElement.nodeName;
"customers"
>>> testDoc.documentElement.getAttribute("rating");
"vip"
>>> testDoc.getElementsByTagName("lastName")[1].firstChild.nodeValue;
"Federer"
>>> var rafie = testDoc.getElementsByTagName("customer")[0];
>>> rafie.getAttribute("id");
"a1234"
>>> testDoc.getElementById("a1234");
null
```
The Element Class

- **getAttribute**
  - Gets value of designated attribute.
  - E.g., if element refers to `<foo bar="a" baz="b">...</foo>`,
    `element.getAttribute("baz")` returns "b"

- **getElementsByTagName**
  - Returns an array of subelements that have this tag name
  - Subelements can be arbitrarily nested

- **hasAttribute**
  - Tests if element has attribute of given name

- **Also inherits from Node class**
  - See next slides
  - All Elements are Nodes, but not vice versa

The Node Class: Properties

- **attributes**
  - An array of the attributes (call `nodeName` on each to get names)

- **childNodes**
  - An array of direct child nodes. 0-length if no children.

- **firstChild, lastChild, parentNode**
  - Specific child nodes. Parent node (null for top element).

- **nextSibling, previousSibling**
  - Related children of the parent node

- **nodeName**
  - For Element nodes, the XML element name

- **nodeType**
  - `Node.ELEMENT_NODE`, `Node.TEXT_NODE`,
    `Node.ATTRIBUTE_NODE`, `Node.CDATA_SECTION_NODE`,
    and a few other options. **Fails on some IE versions!**

- **nodeValue**
  - For Text nodes, the body content.
  - Call normalize first. See next slide.
Fixing Node Types in Internet Explorer

• **Problem**
  – Official standards Node.TEXT_NODE etc., unsupported through Internet Explorer 6 (OK in IE 7)

• **Solution**
  – Redefine them

    ```javascript
    if (!window['Node']) {
      window.Node = new Object();
      Node.ELEMENT_NODE = 1;
      Node.ATTRIBUTE_NODE = 2;
      Node.TEXT_NODE = 3;
      Node.CDATA_SECTION_NODE = 4;
      ...
      Node.DOCUMENT_NODE = 9; ...
    }
    ```

The Node Class: Methods

• **hasAttributes**
  – Does this Node have any attributes at all?

• **hasChildNodes**
  – Does this Node have any children at all?

• **normalize**
  – Merges multiline text nodes.
    • Important if element has body content that spans multiple lines or has extra white space.
    • But you can still have empty text nodes
  – You can call it on root element just once.
    • `xmlDoc.documentElement.normalize();`
Summary

• How to treat the Ajax response as XML
  – var xmlDoc = response.responseXML;
    • For practice and learning, you can also make a string containing XML tags and turn it into an XML document using Firefox or IE-specific functions.
• How to get an array of subelements
  – xmlDoc.getElementsByTagName(…)
• How to get an attribute of an element
  – someElement.getAttribute(…)
• How to get the body content of an element
  – someElement.firstChild.nodeValue