Lecture 17: Ajax
(Asynchronous JavaScript And XML)

Wendy Liu
CSC309F – Fall 2007

Spinning Wheel Progress Indicator

Google Suggest
http://labs.google.com/suggest

Low-level View
The Basics
Overview of Ajax

- It is not a new programming language, but a new way (technique) to use existing standards
- To create better, faster, and more user-friendly and interactive web applications
- Based on JavaScript and HTTP requests
  - Uses JavaScript as its programming language
  - Uses the XMLHttpRequest object to communicate directly with the server
  - Trades data with a web server without reloading the page
- Uses asynchronous data transfer (via HTTP requests) between the browser and the web server
  - Allowing web pages to request small bits of information from the server instead of whole pages
- It is a browser technology independent of web server software

XMLHttpRequest

- JavaScript object
  - Provide two views of the response
    - String (text) and XML
  - Capable of issuing GET, POST, HEAD, PUT, DELETE, OPTIONS requests
- Security limitations apply
  - Same point of origin
    - Can only connect to same domain as currently loaded page
XMLHttpRequest Properties
- Receiving data (handle response)
  - onreadystatechange
  - readyState
  - responseText
  - responseXML

onreadystatechange
- Defines a function to receive data returned by the server after a request is sent
- Must be set before sending request
- The following code defines a function for this purpose (with an empty body for now)
  ```javascript
  var xmlhttp = new XMLHttpRequest();
  xmlhttp.onreadystatechange = function()
  {
    // code for receiving response data
  }
  ```

readyState
- This property holds the status of the server's response
- Each time the readyState changes, the onreadystatechange function will be executed
- State description
  - 0  The request is not initialized
  - 1  The request has been set up
  - 2  The request has been sent
  - 3  The request is in process
  - 4  The request is complete

Update the Function
```javascript
xmlhttp.onreadystatechange=function()
{
  if (xmlhttp.readyState==4)
  {
    // Get the data from the server's response
  }
}
```
**responseText**
- Retrieve text data returned by the server
- **Type:** DOMString (readonly)

```javascript
xmlhttp.onreadystatechange=function()
{
    if (xmlhttp.readyState==4)
    {
        document.getElementById('formentry').value =
            xmlhttp.responseText;
    }
}
```

**responseXML**
- Retrieve document data returned by the server
- **Type:** Document (readonly)

```javascript
var xmldoc=xmlhttp.responseXML.documentElement;

- You can access it as a DOM document
- As you did in A1
```

---

**XMLHttpRequest Methods**
- Asking for data (send request)
  - **open()**
    - Two required arguments
      - method (GET, POST, PUT, DELETE, HEAD, OPTION)
      - server-side URI
  - **send()**
    - One argument
      - data to be sent (DOMString or Document)
        - **null** for GET
        - can be omitted

---

**Execute the Ajax Function**
- Want it to run "behind the scenes"
  ```javascript
  <script type="text/javascript"> 
    function myajax() 
        { . . . /* all of the code from before */ . . . } 
  </script> 
  <form> 
    <input type="text" onkeyup="myajax();" 
        name="userdata" /> 
    <input type="text" id="formentry" /> 
  </form>
  ```
The Basic Ajax Process

- JavaScript
  - Define an object for sending HTTP requests
  - Initiate request
    - Get request object
    - Designate a request handler function
      - Supply as onreadystatechange attribute of request
    - Initiate a GET or POST request
    - Send data
  - Handle response
    - Wait for readyState of 4 and HTTP status of 200
    - Extract return text with responseText or responseXML.
    - Do something with result

Define a Request Object

```javascript
var request;

function getRequestObject() {
  if (window.XMLHttpRequest) {
    return(new XMLHttpRequest());
  } else {
    return(null);
  }
}
```

Browsers that support it

The Basic Ajax Process (cont’d)

- HTML
  - Loads JavaScript
  - Designates control that initiates request
  - Gives ids to input elements that will be read by script

Examples

Putting it Together
Initiate Request

```javascript
function sendRequest() {
    request = getRequestObject();
    request.onreadystatechange = handleResponse;
    request.open("GET", "message-data.html", true);
    request.send(null);
}
```

Handle Response

```javascript
function handleResponse() {
    if (request.readyState == 4) {
        alert(request.responseText);
    }
}
```

Complete JavaScript Code (show-message.js)

```javascript
var request;
function getRequestObject() {
    if (window.ActiveXObject) {
        return(new ActiveXObject("Microsoft.XMLHTTP"));}
    else if (window.XMLHttpRequest) {
        return(new XMLHttpRequest());
    } else {
        return(null);
    }
}
function sendRequest() { // called in XHTML
    request = getRequestObject();
    request.onreadystatechange = handleResponse;
    request.open("GET", "message-data.html", true);
    request.send(null);
}
function handleResponse() {
    if (request.readyState == 4) {
        alert(request.responseText);
    }
}
```

XHTML Code (show-message.html)

```html
<!DOCTYPE html PUBLIC "..." ""
<html xmlns="http://www.w3.org/1999/xhtml">
<head><title>Ajax: Simple Message</title>
<script src="show-message.js" type="text/javascript"></script>
</head>
<body><center>
<table border="1" bgcolor="gray">
<tr><th>Ajax: Simple Message</th></tr>
</table>
<form action="">
<input type="button" value="Show Message" onmouseover="sendRequest()"/>
</form>
</center></body></html>
```
XHTML Code (message-data.html)

Some random message

The Basic Process: Results

Cross-Browser XMLHttpRequest

```javascript
function getRequestObj() {
  var xmlHttp;
  try {   // Firefox, Opera 8.0+, Safari
    xmlHttp=new XMLHttpRequest();
  } catch (e) {   // Internet Explorer
    try {
      xmlHttp=new ActiveXObject("Msxml2.XMLHTTP");
    } catch (e) {
      try {
        xmlHttp=new ActiveXObject("Microsoft.XMLHTTP");
      } catch (e) { alert("Your browser does not support AJAX!");
        return null;
      }
    }
  }
  return xmlHttp;
}
```

High-level View

Frameworks
Motivation

- Challenges of Ajax
  - Browser compatibilities
  - Repeated coding effort

Client-Side Frameworks I

- Rico
  - Designed for drag-and-drop actions, data grids, and what they term cinematic effects (moving widgets, fading a div, and so on)
  - http://opentrico.org

- scriptaculous
  - Animation framework, drag and drop, Ajax controls, DOM utilities, and unit testing
  - http://script.aculo.us/

Client-Side Frameworks II

- Dojo
  - Focused on usability
  - http://www.dojotoolkit.org

- qooxdoo (pronounced [ˈkuːksduː])
  - Sophisticated widgets that allow a thin application to incorporate rich UI features
  - http://qooxdoo.oss.schlund.de

Client-Side Frameworks III

- TIBET
  - Client-side middleware and WYSIWYG tools
  - For constructing web service clients, web portals, and standalone or embedded device web applications
  - Provide support for Web Services, low-level protocols, and pre-built wrappers
    - Google, GMail, Zoe, Amazon, Blogger, Syndic8, Meerkat, XIgnite
  - Fully interactive browser-based IDE that simplifies development, debugging, and unit testing
  - http://www.technicalpursuit.com
Simple Wrappers for XMLHttpRequest
- SACK (Simple Ajax Code Kit)
- XHConn
  - http://xkr.us/code/javascript/XHConn

Client-Side Toolkits
- Yahoo User Interface Library (YUI)
  - Free JavaScript toolkit with some Ajax support
  - http://developer.yahoo.com/yui/
- Google
  - Google Ajax API
    - Search, Feed, Map API
    - http://code.google.com/apis/ajax/
  - Google Mapplet
    - http://code.google.com/apis/maps/documentation/mapplets
  - Google Web Toolkit
    - Write code in Java, translate it to JavaScript
    - http://code.google.com/webtoolkit/

Server-side Framework I
- JSON (JavaScript Object Notation)
  - Text format used to exchange data
    - Like XML, but easier to manipulate
    - http://json.org
- Direct Web Remoting (DWR)
  - Allows Javascript in a browser to interact with Java on a server and helps you manipulate web pages with the results
  - http://getahead.org/dwr/

Server-side Framework II
- SWATO (Shift Web Application TO)
  - Allows you to call server-side Java from a browser
    - A set of reusable and well-integrated Java/JavaScript library
  - Uses JSON to marshal data between the client and server
  - https://swato.dev.java.net/doc/html/
Server-side Framework III
- Ruby on Rails
  - Allow rapid development of Web-based applications
  - Rails has good support for Ajax with several built-in JavaScript libraries that wrap many common features
- http://www.rubyonrails.org

Correction: REST
The original slide is correct

Actions (Verbs)
- Controllers cannot have arbitrary verbs, only these:
  - POST
    - Create - create a new resource
  - GET
    - Retrieve - retrieve a representation of a resource
  - PUT
    - Update - update a resource
  - DELETE
    - Delete - delete a resource