Session 11

Ajax

Reading & Reference

- Reference
  - XMLHttpRequest object
    en.wikipedia.org/wiki/XMLHttpRequest
  - Specification
    developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest
  - JavaScript (6th Edition) by David Flanagan, O’Reilly Press (Available online through Safari textbooks), Section 18.1
  - Internet security same origin policy
    http://en.wikipedia.org/wiki/Same-origin_policy
Learning Goals

- Understand the architecture of Ajax
- Understand the XMLHttpRequest object
- Understand servlet response formats
  - Text
  - Xml
  - Html
  - JSON
- Understand the cross domain issues when requesting servlet data

What is Ajax?

- Asynchronous JavaScript Technology and XML
- Allows incremental update of Web pages within the browser
- Not dependent on any given language or data exchange format, but works well with JavaScript

The XML part of the name is no longer important in Ajax
Ajax Uses

- Real-time form data verification
- Autocompletion
- Filtering & sorting (e.g., sorted table columns)
- Master details (deep tree navigation)
- Expanded user interface controls (e.g., voting)
- Refreshing data on the page (e.g., news/sports)
- Rapid user-to-user communications

Ajax Limitations

- Complexity (development and debugging)
- Not quite standard XMLHttpRequest object
- Viewable source code
- Download of sizeable JavaScript libraries
Classic Browser/Server Interaction

Browser

- Click → "...wait..."
- Click → "...wait..."
- Click

Server

- Processing
- Processing

Ajax Browser/Server Interaction

Browser

- Browser UI
- User events → UI updates

Ajax Engine

- Client engine is key to Ajax model by allowing asynchronous operation

Server

- Processing
- Processing

© Robert Kelly, 2018
XMLHttpRequest Object

- Transport object for communication between client and server
- Methods allow you to
  - Specify request details
  - Extract response data
- Subject to some cross-domain limitations
- No longer a good name since
  - Any text document can be returned (not just XML)
  - Uses https as well as http
  - Handles the response as well as the request

XMLHttpRequest History

- History
  - Microsoft first implemented the XMLHttpRequest object in IE5 for Windows as an ActiveX object.
  - Engineers on the Mozilla project implemented a compatible native version for Mozilla 1.0 (and Netscape 7).
  - Most browsers now support the standard XMLHttpRequest
**Typical Ajax Interaction**

- Client event occurs
- `XMLHttpRequest` object is created
- `XMLHttpRequest` object calls the server (call returns asynchronously)
- Server request is processed by server code (e.g., a servlet)
- Server returns data (e.g., JSON) containing the result
- `XMLHttpRequest` object calls the `callback()` function that processes the result
- The browser document (html) is updated

**Ajax Summary**

- Request to a server is enabled with the `XMLHttpRequest` object, which
  - Specifies the details of the request
  - Can run synchronously or asynchronously
  - Has properties from which you can extract the response
- `XMLHttpRequest` response properties
  - Metadata - status, `statusText`, headers, etc.
  - Text - `responseText` property
  - XML - `responseXML` (Document object)
  - Structured data (e.g., arrays) - sent as text and decoded using `JSON.parse` method

© Robert Kelly, 2018
Common XMLHttpRequest Methods

- `open("method", "URL", [asyncFlag])` - Initializes the request parameters (destination URL, method, and asynchronous flag)
- `send(content)` - Transmits the request, optionally with postable string or DOM object data
- `setRequestHeader("label", "value")` - Assigns a label/value pair to the header to be sent with a request
- `abort()` - Stops the current request
- `getAllResponseHeaders()` - Returns complete set of headers (labels and values) as a string
- `getResponseHeader("headerLabel")` - Returns the string value of a single header label

Common XMLHttpRequest Properties

- `onreadystatechange` - Event handler (function) for an event that fires at every state change
- `readyState` - Object status integer:
  - 0 = uninitialized
  - 1 = loading
  - 2 = loaded
  - 3 = interactive
  - 4 = complete
  - Depends on the content-type of your response
- `responseText` - String version of data returned from server process
- `responseXML` - DOM-compatible document object of data returned from server process
- `status` - Numeric code returned by server (e.g., 404)
- `statusText` - String message accompanying the status code
Example - Ajax Validation

- **Ajax validation with onblur**
  - Empty field responds with a prompt message
  - Data entered is confirmed with a message

---

Ajax Validation in the Servlet

- Validation can be performed with a servlet call

```html
<td>Company <span class="highlight">*</span></td>
<td> <input id="company" name="1" size="35" class="textbox" type="text" onblur="validateCompany();">
  <span>&nbsp;</span>
  <span class="error" id="companyMessage"></span> </td>

HelloAjaxText.html
```

© Robert Kelly, 2018
### JavaScript Ajax Call

```javascript
var req;
var f;
var t;

function validateCompany() {
  f = document.getElementById("companyMessage");
  t = document.getElementById("company");
  + t.value;
  req = new XMLHttpRequest();
  req.open("GET", url, true);
  req.onreadystatechange = companyValidation;
  req.send(null);
}
```

- **Company name sent to server in query string**
- **If the call is asynchronous (true), the callback function must be specified**
- **Note use of a function as first class object**

### Servlet Operation

- **Servlet generates the response**
- **Content-type must be set to MIME type consistent with the output, for example:**
  - text/xml for xml responses
  - text/json for JavaScript object responses
- **Cache control header must be set to "no-cache" (keeps browsers from locally caching responses in which duplicate requests may return different responses)**

```javascript
response.setContentType("...");
response.setHeader("Cache-Control", "no-cache");
response.setHeader("Access-Control-Allow-Origin", "null");
```
Session 11 – Ajax/Servlets

Servlet Code

```java
protected void processRequest(HttpServletRequest request,
HttpServletResponse response)
throws ServletException, IOException {
    response.setContentType("text/plain;charset=UTF-8");
    response.setHeader("Access-Control-Allow-Origin", ";");
    PrintWriter out = response.getWriter();
    String c = request.getParameter("company");
    if (c.equals("")) {
        out.print("Please enter your company name.");
    } else {
        out.print("ok");
    }
}
```

Mime type is plain text

Use the print method to avoid a carriage return/line feed in your response

Header needed if html and servlet are in separate domains

Error message to be displayed in browser

JavaScript Callback Function

```javascript
function companyValidation() {
    if (req.readyState == 4 && req.status == 200) {
        if (req.responseText != "ok") {
            f.innerHTML = req.responseText;
            t.focus();
        } else {
            f.innerHTML = "response is OK";
        }
    }
}
```

f - error message field

t - text box
When an XML Document is Returned

```
function callback() {
    if (req.readyState == 4) {
        if (req.status == 200) {
            // update the HTML DOM based on message
        }
    }
}
```

Use standard JavaScript to modify the DOM document

```
function parseMessage() {
    var message = req.responseXML.getElementsByTagName("message")[0];
    setMessage(message.childNodes[0].nodeValue);
}
```

This is why you should set content-type to text/xml

Notice access into xml tree

When JSON is Returned

```
var xmlhttp = new XMLHttpRequest();
var url = "myTutorials.txt";

xmlhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
        var myArr = JSON.parse(this.responseText);
        myFunction(myArr);
    }
};
```

The JavaScript parse function will convert the text into a JavaScript object
Control in JavaScript

- Notice the transition steps in browser
  - Req.send
  - End of validateCompany
  - Next html statement
  - Callback function (companyValidation)

```javascript
function validateCompany() {
  f = document.getElementById("companyMess");
  t = document.getElementById("company");
  var url = "http://localhost:8080/CSE335-";
  req = new XMLHttpRequest();
  req.open("GET", url, true);
  url = "http:
  req.onreadystatechange = companyValidation;
  req.send(null)
}
```

Screen shots from Chrome JavaScript debugger

Internet Same Origin Policy

- Policy
  - Policy disallows many legitimate requests
  - permits scripts running on pages originating from the same site (scheme, hostname, and port number) - to access each other’s DOM with no specific restrictions
  - Does not allow a page to receive data from a different domain
  - Applies to XMLHttpRequest
- Browsers implement the policy differently (e.g., IE)
- You might encounter a status code of 0 instead of 200 if there is a mismatch of domains
- Domains you might encounter
  - Servlet run in NetBeans – localhost:8080
  - Html opened directly – null
Same Origin Policy Workaround

- New http header
  ```javascript
  resp.setHeader("Access-Control-Allow-Origin", "*");
  ```
  - Supported by most browsers (e.g., Firefox, Safari, and Chrome)
  - Lists origins that may request a file
- Other workarounds available

Workarounds are changing, but the new http header seems to work in most browsers. Not needed if both the requesting page and the servlet reside in the same Web App.

Comment on readyState Property

```javascript
if (req.readyState==4 & req.status==200){
  ...
}
```

- Not good style to test hard-coded values (i.e., magic numbers) of readyState
- Constants available in XMLHttpRequest object, but not implemented uniformly in browsers
  - 0 - UNSENT
  - 1 - OPENED
  - 2 - HEADERS_RECEIVED
  - 3 - LOADING
  - 4 - DONE

You will most likely use higher level libraries that will make this invisible.