Introduction to AJAX

Pat Morin

COMP 2405

Outline

- What is AJAX?
  - History
  - Uses
  - Pros and Cons
- An XML HTTP Transaction
  - Creating an XMLHTTPRequest
  - Setting up an XMLHTTPRequest
  - Sending an XMLHTTPRequest
- Receiving an XML reply

Carleton UNIVERSITY
Canada's Capital University

What is AJAX?

- AJAX stands for Asynchronous Javascript And XML
- AJAX is not a programming language
- AJAX is a way of using existing standards (JavaScript and XML) to make more interactive web applications
- AJAX was popularized in 2005 by Google (with Google suggest)

Carleton UNIVERSITY
Canada's Capital University

An AJAX Application

- Recall the standard HTTP transaction
  - 1. Client opens connection to server
  - 2. Client sends request to server
  - 3. Server sends reply to client
  - 4. Client and server close connection
- After Step 4, the client renders the document and this may include running some JavaScript
- In an AJAX application, the JavaScript code then communicates with the server *behind the scenes*

Carleton UNIVERSITY
Canada's Capital University

An AJAX Application (Cont'd)

- Communication with the server takes place asynchronously, and transparently to the user
- Data is exchanged with the server without the need for a page reload
- This is accomplished through a special kind of HTTP request

Carleton UNIVERSITY
Canada's Capital University

Typical AJAX Event

- A typical AJAX transaction looks like this:
  1. User triggers some event (presses a key, moves mouse, ...)
  2. Event handler code sends HTTP request to server
  3. Server replies triggering code on client
  4. Reply handler code updates web page using server's reply
- Between steps 2 and 3 the web page is still usable (event is asynchronous)
- At no point during the transaction does the browser open a new web page

Carleton UNIVERSITY
Canada's Capital University

Carleton UNIVERSITY
Canada's Capital University

Pros and Cons of AJAX

- Pros:
  - Allows web applications to interact with data on the server
  - Avoid clunky GET/POST send/receive interfaces – web apps look more and more like real applications
  - Some applications can only be realized this way
    - Eg: Google Suggest offers interactive access to one of the largest data collections in the world
  - For office style applications, user's data is stored on a reliable server, accessable from any web browser
- Cons:
  - Tough to make compatible across all browsers
  - Should have a low-latency connection to the server
  - Can be server intensive
    - Eg: Google Suggest generates a search for every keystroke entered

Setting up an AJAX Transaction

- Create an `XMLHTTPRequest` object
- Set up the request's `onreadystatechange` function
- Open the request
- Send the request

Canada's Capital University

Creating an XMLHTTPRequest Object

```
function sendRequest()
  var xmlHttp = GetXmlHttpObject();
  if (!xmlHttp) {
    return false;
  }
  xmlHttp.onreadystatechange = function() {
    if (xmlHttp.readyState == 4) {
      alert("Request complete");
    }
  }
  var requestURI =
    "http://myserver.org/somepage.txt";
  xmlHttp.open("GET", requestURI, true);
  xmlHttp.send(null);
}
```

Carleton UNIVERSITY
Canada's Capital University

The XMLHTTPRequest Object

- An `XMLHTTPRequest` object is in one of 5 states, as indicated by the `readyState` property
    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
- Every time the readyState property changes the `onreadystatechange` property (a function) is called

Carleton UNIVERSITY
Canada's Capital University

Setting onreadystatechange

```
function sendRequest()
  var xmlHttp = GetXmlHttpObject();
  if (!xmlHttp) {
    return false;
  }
  xmlHttp.onreadystatechange = function() {
    if (xmlHttp.readyState == 4) {
      alert("Request complete");
    }
  }
  var requestURI =
    "http://myserver.org/somepage.txt";
  xmlHttp.open("GET", requestURI, true);
  xmlHttp.send(null);
}
```

Carleton UNIVERSITY
Canada's Capital University

Carleton UNIVERSITY
Canada's Capital University

The open and send functions

- The open function of an XML HTTP request takes three arguments
  - `xmlHttp.open(`*`method`*`, `*`uri`*`, `*`async`*`)`
  - *`method`* is either `"GET"` or `"POST"`
  - *`uri`* is the (relative) URI to retrieve
  - *`async`* determines whether to send the request asynchronously (`true`) or synchronously (`false`)
  - The domain of the *`uri`* argument must be the same as the domain of the current page
- The send function takes one argument
  - `xmlHttp.send(`*`content`*`);`
  - *`content`* is the content to send (useful when *`method`*=`"POST"`)

Sending the Request

```
function sendRequest()
  var xmlHttp = GetXmlHttpObject();
  if (!xmlHttp) {
    return false;
  }
  xmlHttp.onreadystatechange = function() {
    if (xmlHttp.readyState == 4) {
      alert("Request complete");
    }
  }
  var requestURI =
    "http://myserver.org/somepage.txt";
  xmlHttp.open("GET", requestURI, true);
  xmlHttp.send(null);
}
```

Carleton UNIVERSITY
Canada's Capital University

The responseText Property

- When an XMLHTTPRequest is complete (`readyState == 4`) the `responseText` property contains the server's response, as a String

Carleton UNIVERSITY
Canada's Capital University

Example Code (Client Side)

```
function sendRequest(textNode)
  var xmlHttp = GetXmlHttpObject();
  if (!xmlHttp) {
    return false;
  }
  xmlHttp.onreadystatechange = function() {
    if (xmlHttp.readyState == 4) {
      textNode.nodeValue =
           xmlHttp.responseText;
    }
  }
  var requestURI =
    "http://greatbeyond.org/cgi-bin/request.cgi";
  xmlHttp.open("GET", requestURI, true);
  xmlHttp.send(null);
}
```

Carleton UNIVERSITY
Canada's Capital University

Example Code (Server Side)

- And we might have the following `request.cgi` in the `cgi-bin` directory of `greatbeyond.org`

```
#!/usr/bin/perl

print("Content-type: text/plain\n\n");
print("57 channels and nuthin' on");
```

Carleton UNIVERSITY
Canada's Capital University

Some Notes

- An XMLHTTPRequest object can send the request to any URI as long as it has the same domain as the page that requests it
- This URI can refer to a CGI script or even just an HTML document
- Note the big security risk for the client
  - JavaScript can send anything to the server
  - Client needs to restrict what JavaScript has access to
- This is still not AJAX
  - Where's the XML?

Carleton UNIVERSITY
Canada's Capital University

Putting the X in AJAX

- The X in AJAX comes from XML
- In an XML HTTP request, we usually expect the server to respond with some XML
- What is XML?
- Short answer: like HTML but
  - You can make up your own tag names
  - All tags have to be closed (and there is a shorthand)
- Long answer: will have to wait

Carleton UNIVERSITY
Canada's Capital University

An Example XML File

- Notice
  - the new tags (we just made them up)
  - An XML version number
  - One tag contains everything (and becomes the root of the document tree)

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<note>
 <to>Tove</to>
 <from>Jani</from>
 <heading>Reminder</heading>
 <body>Don't forget me this weekend!</body>
</note>
```

Carleton UNIVERSITY
Canada's Capital University

Why Respond with XML?

- We can look at the XML text within a response using the `responseText` property
- Even better, we can use the `responseXML` property to access the XML
- Best, `responseXML.documentElement` contains the document tree for the XML
- This is a document tree in the DOM model that we've seen before (just like `document`)

Carleton UNIVERSITY
Canada's Capital University

Example

```
function sendRequest() {
  var xmlHttp = GetXmlHttpObject();
  if (!xmlHttp) {
    return false;
  }
  xmlHttp.onreadystatechange = function() {
    if (xmlHttp.readyState == 4) {
      var xmlDoc =
        xmlHttp.responseXML.documentElement;
    }
  }
  var requestURI = xmlURI;
  xmlHttp.open("GET", requestURI, true);
  xmlHttp.send(null);
}
```

Carleton UNIVERSITY
Canada's Capital University

Summary

- An AJAX transaction involves the client sending an asynchronous HTTP request and the server responding with XML
  - The client processes the resulting XML document tree
- AJAX applications run entirely on the client except when they need to access data on the server
  - Can treat the server as a database/file system
- Well-written AJAX applications, running with a fast Internet connection, can be as nice to use as traditional applications (or nicer)

Carleton UNIVERSITY
Canada's Capital University