Asynchronous JavaScript And XML (AJAX)

Hamid Zarrabi-Zadeh

Web Programming – Fall 2013
Outline

• History
• What is AJAX?
• Using AJAX
• A Live Example
• jQuery AJAX
• Summary
History

- In the 1990s, most web sites were based on complete HTML pages
- Each user action required that a new page (or the same page) be loaded from the server
- This process was inefficient
  - it was slow and required user input
- For web-based email this meant that users had to manually reload their inbox to check and see if they had new mail
In 1999, Microsoft created the XMLHttpRequest ActiveX control in IE5, allowing the browser to communicate with the server without requiring a page reload.

The idea was later adopted by Mozilla, Safari, Opera and other browsers as the XMLHttpRequest JavaScript object.

The term AJAX was coined in 2005 by Jesse James Garrett in an article entitled "Ajax: A New Approach to Web Applications"
What is AJAX?

• AJAX stands for Asynchronous JavaScript And XML
• It is a way of using existing standards (such as JavaScript and XML) to make more interactive web applications
• AJAX was popularized in 2005 by Google, specially, with Google suggest
A typical AJAX transaction looks like this:
1. An event is triggered by the user/browser (such as mouse events, keyboard events, or time events)
2. Event handler sends an HTTP request to server
3. Server replies to the request
4. The reply handler 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
AJAX Pros

- AJAX allows web applications to exchange data with the server without the need for a page reload.
- Communication with the server takes place asynchronously, and transparently to the user.
- AJAX allows us to avoid clunky GET/POST send/receive interfaces.
- Many web applications can only be realized this way (e.g., Google Maps, Google suggest, etc.).
AJAX Cons

• Different browsers implement the AJAX differently
• AJAX can be server intensive
  – e.g., Google Suggest generates a search for every keystroke entered
• Needs a low-latency connection to the server
• Dynamic web page updates are difficult to bookmark, and to be indexed by search engines
Using AJAX
Transaction Steps

• Create an XMLHttpRequest object
• Set up the response handler
• Open the request
• Send the request
Sample AJAX

- Here is a simple AJAX transaction:

```javascript
var httpRequest = new XMLHttpRequest();

httpRequest.onreadystatechange = function() {
  if (httpRequest.readyState == 4) {
    alert('Request complete!');
  };

httpRequest.open('GET', 'something.py', true);
httpRequest.send(null);
```
Open and Send Methods

• The `open` method takes three arguments:
  ```javascript
  httpRequest.open(method, url, async)
  ```
  – `method`: 'GET' or 'POST' (or any HTTP request)
  – `url`: the (relative) URL to retrieve
  – `async`: determines whether to send the request asynchronously (true) or synchronously (false)

• The `send` method takes one argument:
  ```javascript
  httpRequest.send(content)
  ```
  – `content`: the content to send (useful when method='POST')
Ready State

- The `readyState` property holds the status of the XMLHttpRequest:
  - 0: request not initialized
  - 1: server connection established
  - 2: request received
  - 3: processing request
  - 4: request finished and response is ready
Handling Response

- When an XMLHttpRequest is complete (readyState == 4) the status property contains the response code of the HTTP response

```javascript
if (httpRequest.status === 200) {
    // perfect!
} else {
    // there was a problem with the request,
    // e.g., the response may be 404 (Not Found)
    // or 500 (Internal Server Error)
}
```
Accessing Response

• After checking the HTTP response code, we can access the data through:
  – `responseText` – the server response as a string
  – `responseXML` – the response as an XMLDocument object that you can traverse using the JavaScript DOM functions
Example

- This is a simple code for parsing an xml file

```javascript
xmlDoc = httpRequest.responseXML;
books = xmlDoc.getElementsByTagName('title');
res = '';
for (var i = 0; i < books.length; i++)
    res += books[i].firstChild.nodeValue + '<br>';
document.getElementById('books').innerHTML = res;
```
A Live Example: Suggestions
<!DOCTYPE html>
<html>
<body>

<h2>Type a name:</h2>

<form action="">
  <input type="text" onkeyup="suggest(this.value)">
</form>

<p>Suggestions: <span id="names"></span></p>

</body>
</html>
var xhr;

function suggest(str) {
    if (str.length == 0) {
        document.getElementById('names').innerHTML = '';
        return;
    }
    xhr = new XMLHttpRequest();
    xhr.onreadystatechange = handler;
    xhr.open('GET', 'suggest.py?q=' + str, true);
    xhr.send(null);
}

function handler() {
    if (xhr.readyState == 4 && xhr.status === 200)
        document.getElementById('names').innerHTML = xhr.responseText;
}
import cgi

form = cgi.FieldStorage()
q = form.getvalue('q').lower()

names = ['Ali', 'Ahmad', 'Azin', 'Bahram', 'Baran', 'Zahra']

res = []
for name in names:
    if name.lower()[0:len(q)] == q:
        res.append(name)

print('Content-type: text/html\n')
print(', '.join(res))
jQuery and AJAX

- jQuery provides AJAX support that abstracts away browser differences
- It offers both a full-featured $.ajax() method, and simple convenience methods such as $.get(), $.getScript(), $.getJSON(), $.post(), and $.load()

```
$.get('foo.py', function(response) {
    console.log(response);
});
```

```
$('#names').load('suggest.py', {q: str});
```
Core Ajax

- $.ajax() is the core of AJAX support in jQuery
- It takes a configuration object that contains all instructions required to complete the request

```
$.ajax({
    url: 'test.py',
    data: { id: 110 },
    type: 'GET',
    dataType: 'json', // response data type

    success: function(json) {
        $('#title').text(json.title);
    },
    error: function(xhr, status) {
        alert('Sorry, there was a problem!');
    }
});
```
The types of data that you may expect back from the server include:
- xml
- html
- text
- script
- json
- jsonp
The following callback function can be set:
- `beforeSend` – called before sending request
- `error` – invoked, if the request fails
- `success` – is invoked, if the request succeeds
- `dataFilter` – is invoked immediately upon successful receipt of response data
- `complete` – fires, when the request finishes, whether in failure or success
Status Code

- You can specify callback functions for any specific HTTP response code

```javascript
$.ajax({
  url: 'test.py',
  statusCode: {
    404: function() {
      alert('Page not found');
    }
  }
});
```
In general, AJAX requests are limited to the same domain as the page making the request.

We can avoid this limitation with JSONP.

```javascript
$.ajax({
    url: 'http://example.com/get.py',
    dataType: 'jsonp',
    jsonp: 'callback'
});
```
Sending Form Data

- You can use `serialize()` method to serialize form inputs before posting to the server

```javascript
$.ajax({
    url: 'get-form.py',
    type: 'POST',
    data: $('#form').serialize(),
    success: function(data) {
        console.log(data);
    }
});
```
Summary

• An AJAX transaction involves the client sending an asynchronous HTTP request and the server responding with some data.

• AJAX applications run entirely on the client except when they need to access data on the server.

• Libraries like jQuery make working with AJAX easier and cross-browser.
References

- jQuery AJAX
  - http://stage.learn.jquery.com/ajax/
- Mozilla Developer Network AJAX Page
- Internet Programming by Pat Morin
  - http://cg.scs.carleton.ca/~morin/teaching/2405