

# Asynchronous Web Applications With Ajax

**Jeff Offutt**

<http://www.cs.gmu.edu/~offutt/>

**SWE 432**

**Design and Implementation of  
Software for the Web**

## Motivation

- Synchronous vs. asynchronous
  - A phone call is synchronous – both parties have to be on the phone at the same time
  - A text message is asynchronous – one party sends a message and the other can retrieve it later
    - Neither party has to wait for the other's response
- The web has the potential for fully distributed applications
  - They can run synchronously or asynchronously
- The request / response cycle used in most web applications makes all communication synchronous
  - The client has to wait for the server to respond

**This eliminates one of the most powerful aspects of  
distributed programming !**

## Ajax History

- Ajax uses Javascript to allow asynchronous interaction between the client and the server
  - Users do not need to click “submit”
  - Often used to respond to events in the UI
- History
  - HTML iframe, from Netscape 4 and IE4, can send asynchronous requests
  - Microsoft introduced XmlDocument and XMLHttpRequest to make asynchronous requests
- Early major uses : Google Maps and Google Mail

**Goal is to improve usability by allowing web apps to respond in ways that look more like desktop apps**

10/18/2011

© Ofutt

3

## Ajax Approaches and Technology

- Two important characteristics
  1. Client requests handled asynchronously
  2. Client modifies only small parts of the current document
- Ajax stands for “Asynchronous Javascript and XML”
  - Client : JavaScript, XML, XMLHttpRequest, DOM, CSS
  - Server : Any web app technology (servlets, JSP, PHP, ASP.NET)
- Ajax currently uses the XMLHttpRequest object
- Lots of frameworks and toolkits now used to create Ajax applications
  - Prototype, Dojo, JavaServerFaces, Rails, ASP.Net Ajax, ...

10/18/2011

© Ofutt

4

## Ajax Overview

- Example application
  - Help users fill in a form
  - Zip code, city, state ... when a zip code is entered, the client asks the server for the probable city and state
  - JS used to put the response into the form
- Form
  - Reference the JS source file in its head
  - Must register an event handler on the blur event in the zip code text box
- JS must have a blur handler and a response handler

<http://www.cs.gmu.edu/~offutt/classes/432/slides/ajax-sebestaCh10source/popcornA.html>

10/18/2011

© Offutt

5

## Ajax Example–Request Phase

- Client communicates to the server with the XMLHttpRequest object

```
var xhr = new XMLHttpRequest ();
```
- Server returns a sequence of notices, or *callbacks*, to the client (0, 1, 2, 3, 4)
  - 4 indicates the response is complete
- The callbacks call the response function
  - Response function must be registered in onreadystatechange property of the XMLHttpRequest object

```
xhr.onreadystatechange = receivePlace;
```

10/18/2011

© Offutt

6

## Ajax Example–Request Phase (2)

- The handler then calls the open method of the xhr object
- Parameters :
  - HTTP method (GET or POST)
  - URL of the response component on the server
  - A boolean literal to indicate if the request is asynchronous (true) or synchronous (false)
  - The form data must be attached to the URL if GET is used  
`xhr.open ("GET", "getCityState.php?zip=" + zip, true);`
- The response component must be on the same server as the original HTML
- Request is sent with the send method  
`xhr.send (null);`

<http://www.cs.gmu.edu/~offutt/classes/432/slides/ajax-sebestaCh10source/popcornA.js>

10/18/2011

© Offutt

7

## Ajax Example–Request Phase (3)

- Response component returns data in response to the request from the JS
- Sebesta's example uses PHP
  - Response data is produced with a print statement
- In a servlet, we implement the `doGet()` method and put the response in the usual Response object
  - We do not need to send an entire HTML page, just a string
  - In fact, we do not need to call `setContentType()`
  - A security rule requires that the response servlet be on the same server as the original document

<http://www.cs.gmu.edu/~offutt/classes/432/slides/ajax-sebestaCh10source/getCityState.php>

10/18/2011

© Offutt

8

## Ajax Example – Receiver Phase

- When the response component on the server finishes, it
  1. invokes the specified callback function
  2. sends the response object to the client
- The callback function is a JS with no parameters
  - It needs to access the XMLHttpRequest object
  - If the object is global, simultaneous requests and responses could cause concurrency conflicts (remember ... asynchronous !)
  - This example registers the code, not just the function name

<http://www.cs.gmu.edu/~offutt/classes/432/slides/ajax-sebestaCh10source/popcornA.js>

10/18/2011

© Offutt

9

## Ajax Return Document Options

- This example shows the response as a string
- Ajax server software components can also return
  - XHTML
  - XML
  - Javascript Object Notation (JSON)

10/18/2011

© Offutt

10

## Ajax Security Issues

- If you have security checks in the JS or HTML, remember that users can modify that code
  - Security must be duplicated on the server
- Ajax applications have many small server-side programs, increasing the attack surface of the entire application
- Servers that provide JS as a response open themselves up to cross-site scripting attacks

10/18/2011

© Ofutt

11

## Ajax Summary

- Asynchronous interaction provides for a much richer user experience
- Despite initial concerns, performance is extraordinary
- Leveraging existing technologies was brilliant
  - Any server-side software technology can be used
- Adding Ajax capability is fairly simple—the interesting part is imagining what we can do with it
- Puts more emphasis on knowing Javascript—we have to use the response text

10/18/2011

© Ofutt

12

# Ajax Examples

Most of the previous examples are fully worked out here:

<http://www.cs.gmu.edu/~offutt/classes/432/examples/ajax/>