



**INTERTECH TRAINING**

# **AJAX – GETTING REAL**

**INTERTECH TRAINING  
MAY 31, 2006**



## Objectives

- Explore Ajax, and discover the technologies behind the hype – Getting Real
- Take a look at some popular Ajax enabled sites and see some code to get started
- Discuss some potential gotcha's and how to alleviate or remedy them
- Learn a few techniques to help make the most of performance, both client and server side
- See emerging Ajax design patterns
- Discuss arguments against Ajax
- Share some good Ajax resources



## Intended Audience

- Web developers
- Software developers
- Team leads
- Managers
- Those interested in Ajax and its potential uses



# A Few Opening Thoughts

- Ajax is a hot topic right now
  - There are many who view Ajax as a bunch of hype
  - There are many who have been using Ajax for years, and don't appreciate the latest buzzword
  - The hype can sometimes cloud the reality
- Effort has been made to try and ensure that the Ajax solutions and examples presented will be cross-browser compatible
- Most Ajax developers tend towards implementations that work rather than sticking with standards

<note>While a fan of standards and the DOM, I have opted to use the innerHTML property for the purposes of example. See <http://domscripting.com/blog/display/35> for an interesting discussion.</note>



## Outline

- Ajax – What is it?
- Ajax – What is it *Really*?
- History and Background
- Demonstrations and Code Examples
- Toolkits
- Potential Hazards – Avoid, Obviate, Alleviate
- Optimization Techniques and Tips
- Design Patterns
- Looking Ahead



## Ajax – What is It?

- Ajax
  - Asynchronous
  - Javascript
  - And
  - XML
- Term coined in article written by Jesse James Garrett of Adaptive Path, February 18, 2005



# Ajax – What is It?

- Improved interactivity with the web user
- An approach focusing on user interface aspects of development
- Enhances the web experience, giving web applications more muscle and flexibility in design as traditional software developers have had for years
- The silver bullet



# Ajax – What is It *Really*?

- An approach to web development using a host of available tools, techniques, and tricks
  - Javascript (ECMAScript) code is the muscle on the client side
  - The browsers DOM (Document Object Model) is used for interactivity with elements of the browser and web page
  - MSXML's XMLHTTP or XMLHttpRequest for client/server communications
  - CSS/DHTML for user interactivity and interface changes
  - XML for packaging data sent between client and server (If it makes sense - JSON (Javascript Object Notation) and plain text are alternatives)



## History and Background

- Microsoft's Innovations
  - Remote Scripting (1998): Using Applet or Active X (MSXML)
  - Scriptlets
  - Data Islands
  - IFrame (IE 3, 1996)
- Microsoft Web Exchange first "Ajax" implementation



# History and Background

- Around this same time, DHTML was also introduced
  - Browser DOM
  - Javascript
  - CSS
- DHTML techniques were used, but could not really live up to full potential...



## History and Background

- Why didn't Remote Scripting or DHTML really catch on?
  - Browser competition and differences
  - Web was still relatively young
  - No widely available "killer app"
  - Javascript is not a full-fledged programming language



## History and Background

- Tipping point in 2005
  - Article by Jesse James Garrett of Adaptive Path gave these technologies a fancy name
  - Netscape's demise and Internet Explorer dominance
  - Stronger standard definitions: DOM, ECMAScript, XML
  - Firefox, upcoming Internet Explorer, etc. implementations more closely fit standards
  - An online marketplace ready for more
  - Killer apps...

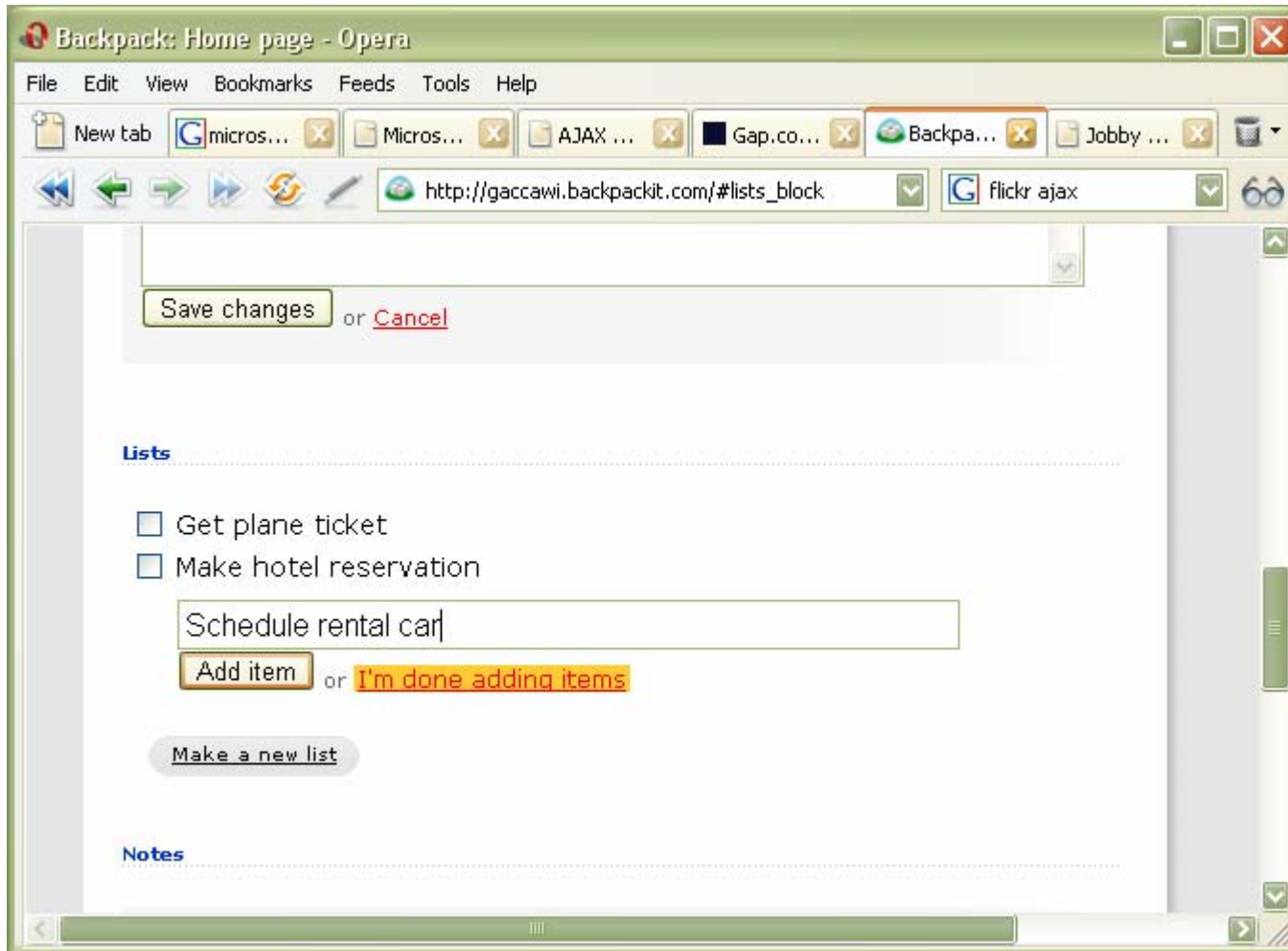


## Demonstration and Sample Code

- Some killer apps
  - Backpack
  - GMail
  - Flickr
  - Google Suggest
  - Google Maps
  - Many, many more new apps every day!



## Demonstration and Example Code





## Demonstration and Sample Code


Google - Opera

File Edit View Bookmarks Feeds Tools Help

New tab [G] microsoft groove - Google... [X] [G] Microsoft Outlook Web Ac... [X] [G] Google [X]

http://www.google.com/webhp?complete=1&hl=en [G] Google

Personalized Home | Sign in



Web Images Groups News Froogle Maps more »

ajax

ajax	3,840,000 results
ajanta	190,000 results
ajanta caves	46,600 results
ajax ontario	275,000 results
ajax grips	8,860 results

As you type, Google suggests:

Advanced Search  
Preferences  
Language Tools

results. [Learn more](#)

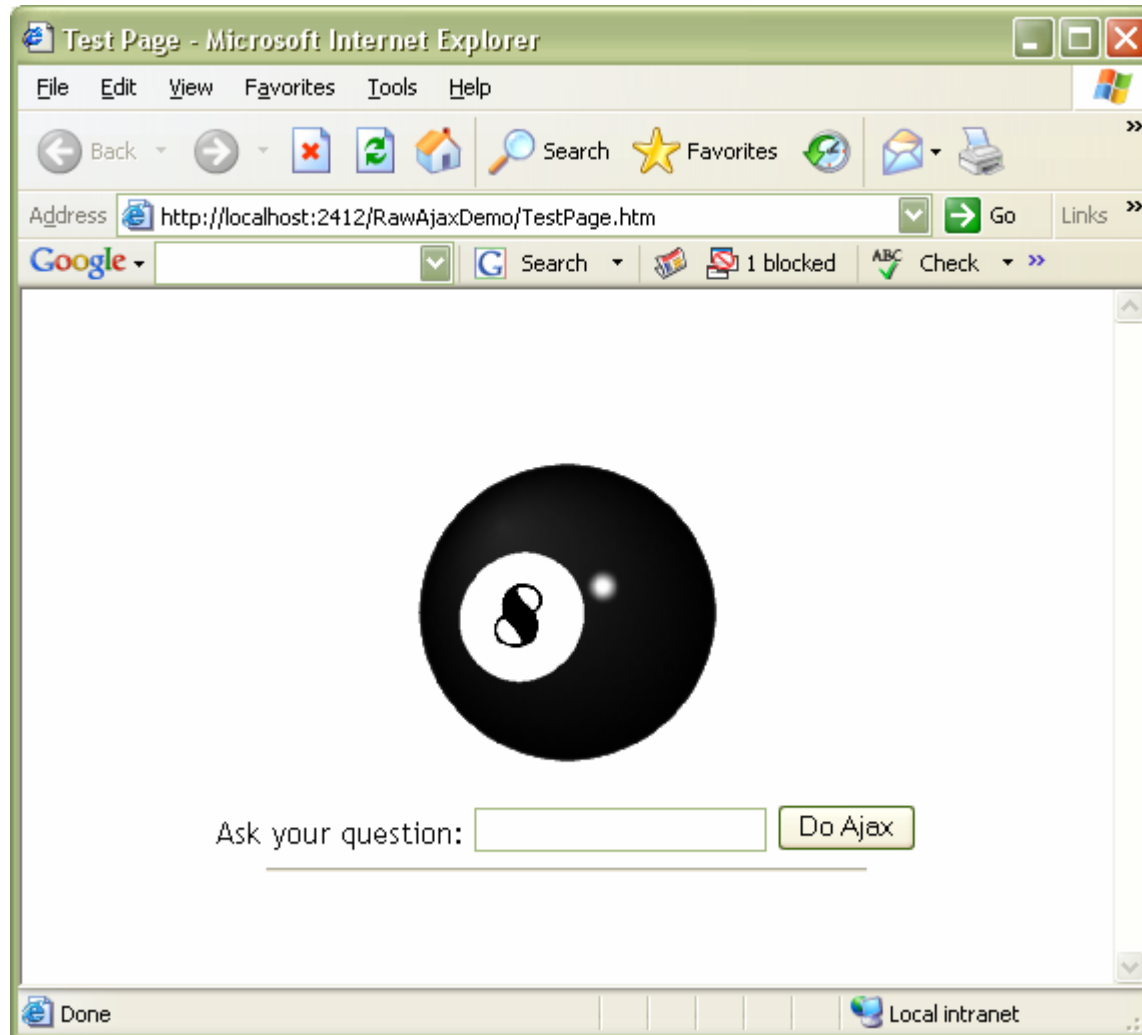


## Demonstration and Sample Code

The screenshot shows a web browser window titled "10 market st, san francisco - Google Maps - Opera". The browser's address bar contains the URL "http://www.google.com/maphp?hl=en&tab=wl&q=" and the search term "10 market st, san francisco". The Google Maps interface is visible, featuring the Google logo, navigation links (Web, Images, Groups, News, Froogle, Maps, more), and a search bar. Below the search bar, the address "10 Market St, San Francisco, CA 94111" is displayed in a popup window. The popup also includes a link to "Make this my default location" and directions links "To here" and "From here". The map shows a street view of San Francisco, with a red pin marking the location of 10 Market St. The popup window has a close button (X) and a "Map" button. The map interface includes navigation controls (directional arrows, zoom in/out, pan) and a "Print" button. The bottom of the map shows the copyright notice "©2006 Google - Map data ©2006 NAVTEQ™ - Terms of Use".



## Demonstration and Example Code





## Demonstration and Code Example

```
<script type="text/javascript">
var xmlhttp=false;
function doAjax () {
    try {
        xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
    }
    catch (e) {
        xmlhttp = false;
    }
    if (!xmlhttp && typeof XMLHttpRequest != "undefined") {
        try {
            xmlhttp = new XMLHttpRequest();
        }
        catch (e) {
            xmlhttp = false;
        }
    }
    <!-- snip -->
}
```



## Demonstration and Code Example

```
if (xmlhttp) {
    xmlhttp.onreadystatechange=checkData;
    xmlhttp.open("POST", "MagicWS.asmx/TellFortune", true);
    xmlhttp.setRequestHeader("Host", "localhost");
    xmlhttp.setRequestHeader("Content-Type",
"application/x-www-form-urlencoded");
    var s = "questionAsked=" + SearchKey.value;
    xmlhttp.setRequestHeader("Content-Length", s.length);
    xmlhttp.send(s);
}
function checkData() {
    if (xmlhttp.readyState == 4)
        document.getElementById("answer").innerHTML =
xmlhttp.responseText;
}
```



## Toolkits

- There are a large number of toolkits available
  - As of Tuesday May 22, 2006 there were 134
  - Every day it seems like more are added...
- The toolkits can be categorized as
  - Pure Javascript libraries
  - Kits for particular server environments
  - Kits for particular programming languages



## Toolkits

- For .NET, two popular ones are
  - Atlas
  - Ajax.Net
- There are a number of general Javascript toolkits available
  - Dojo: <http://dojotoolkit.org>
  - Backbase: <http://www.backbase.com>
- Microsoft provides the Atlas toolkit:  
<http://atlas.asp.net>



## Toolkits

- Primary reasons to use a toolkit
  - Avoid Javascript and use familiar object-oriented programming language and techniques
  - Guaranteed cross-browser implementation
  - Speed development time
  - Reduce maintenance time
- Primary features of toolkits
  - API
  - Interfaces to use or classes to derive and extend
  - Javascript libraries of functions
  - Support via strong user community



## Toolkits – Atlas Example

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeFile="Default.aspx.cs" Inherits="_Default" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">

<!-- In our case, we are calling an asmx. -->
<!-- Add the following reference to include it -->

<atlas:ScriptManager runat="server" ID="ServiceReference">
  <Services>
    <atlas:ServiceReference Path="~/MagicWS.asmx" />
  </Services>
</atlas:ScriptManager>
</head>
```



## Toolkits – Atlas Example

```
<body><form id="form1" runat="server"><div>
<!-- Include a text box to obtain the query string -->
Ask your question: <input type="text" id="SearchKey"/>

<!-- Now, associate a custom Javascript function with -->
<!-- desired event. In our case, a button click. -->
<input id="SearchButton" type="button" value="search"
onclick="DoSearch()" />
</div></form>
<!-- snip -->

<!-- Include a div/span to hold the results from the web -->
<!-- service call. -->
<div><span id="Results"></span></div>
```



## Toolkits – Atlas Example

```
<script type="text/javascript">
<!-- Utilize web service "proxy" class, calling method -->
<!-- passing argument, and indicate the function to -->
<!-- handle the result. -->
    function DoSearch() {
        var SrchElem = document.getElementById("SearchKey");
        MagicWS.TellFortune(SrchElem.value, OnRequestComplete);
    }
<!-- Using the DOM, get result node. Set the innerHTML. -->
    function OnRequestComplete(result) {
        var RsltElem = document.getElementById("Results");
        RsltElem.innerHTML = result;
    }
</script>
</body>
</html>
```



## Potential Hazards

- A list of Ajax mistakes was started by Alex Bosworth, and is maintained/updated collaboratively
  - Includes development specific traps
  - Includes some items that may be missed or overlooked
- Let's take a look...



## Potential Hazards

- ✘ Using Ajax for the sake of using Ajax
  
- ☑ Remember careful design principles and include Ajax only where it makes sense for your application



# Potential Hazards

- ✘ Failing to visually indicate Ajax activity
  - Such as busy on status bar, “working” graphic, mouse change, etc.
  
- ☑ Use graphics, status bar, etc. to indicate “busy”



## Potential Hazards

- ✘ Breaking the back button
  - Users often rely on the back button
  - This also includes breaking browser history, and page URL bookmarks
- ☑ Some toolkits make it easy to provide a callback to catch the back button click (Dojo and Backbase provide ways)
- ☑ There are some “hacks” involving IFrames, #Hashs, location.href and the like, Google back+button+Ajax
- ☑ Don't worry about the back button
- ☑ Only use Ajax in ways not expecting navigation
- ☑ Ajax is for *applications* not *sites*, meaning a paradigm shift
- ☑ Provide alternate traditional pages with same functionality



## Potential Hazards

- ✘ Changing state when doing a “GET”
  - Violates expected behavior due to HTTP guidelines
  
- ☑ Use POST when setting data
- ☑ Carefully design application to use GET only when retrieving data



## Potential Hazards

- ✘ Leaving people behind
  - Offline
  - Mobile users
  - Non-JS enabled browsers
  - Web accessibility (blind, hearing impaired, etc.)
- ☑ Provide alternative, traditional implementations when possible
- ☑ Utilize Ajax for performance improvements, general ease of use improvements, but not for core functional aspects of a site



## Potential Hazards

### ✘ Slowness

- Ajax by definition means improved performance, but poor design can counteract the benefits
- Too much/complex client side logic
- Hitting the server too often
- Sending too much data back and forth

☑ Careful design

☑ Sensible use of bandwidth



## Potential Hazards

- ✖ Failure to accommodate browsers in target market
  - Differences in DOM implementations
  - Differences in ECMAScript implementations
  - CSS/DHTML differences
- ☑ Test in all target environments
- ☑ Utilize standards where possible to minimize future changes to code
- ☑ Utilize Ajax for bells and whistles rather than core functionality
- ☑ Use toolkits with cross-browser implementations



## Potential Hazards

- ✘ Overlooking security
  - Sending data in clear text
  - Sending form data without preprocessing
- ☑ Encryption and https where needed
- ☑ Be mindful of security when designing



## Potential Hazards

- ✘ Unexpected changes to the web page
  - Scrolling or additional content above may move cursor location within the page
  - Automatic reactive state changes, like to checkboxes, may be too much activity and compromise logical flow
- ☑ Use careful design, add content below cursor, in expandable sections, etc.
- ☑ Make batch UI changes when logically grouped to minimize “noise”



## Potential Hazards

- ✘ Blocking Spidering
  - When an app has large amounts of text without page reload there are bound to be trouble with search engines
  
- ☑ If your site needs to be detected by spiders, you should ensure that you are not making your pages too “heavy”



## Potential Hazards

- ✘ Neglecting other areas of the page
  - Outside the Ajax “area” there may be text, graphics, etc. on the page needing update after making a call
  
- ☑ Careful design can resolve this issue
  
- ☑ Rigorous testing may be needed to reveal these overlooked details



## Potential Hazards

- ✘ Character set may cause poor or incorrect rendering of text content
  - XMLHttpRequest posting uses UTF-8 regardless of the character set for the container page
- ☑ Utilize UTF-8 or UTF-16
- ☑ Explicitly set character sets for the browser using meta http-equiv tag
- ☑ In posts using content-type header
- ☑ In databases or other data sources
- ☑ In HTTP response header



# Potential Hazards - Development

- ✘ XMLHttpRequest is not cross-domain
  - You cannot send an Ajax request to a location outside the domain of the requester page
  
- ☑ Application proxy, like a same domain web service intermediary to outside domain
- ☑ Use alternatives such as script tag tricks or IFrames, and not the XMLHttpRequest at all
- ☑ Server configuration for proxying



# Potential Hazards - Development

- ✘ Browser differences still exist
  - While Internet Explorer 6 is the most common browser by a good margin, there are still other browsers out there
  - Internet Explorer 7 beta is out, and this will be another browser version to accommodate
  
- ☑ A primary benefit to using a toolkit is that they resolve this issue
  
- ☑ Be mindful of differences and code around them, testing in all target browsers to ensure functionality



# Potential Hazards - Development

- ✘ Asynchronous request handling means there isn't any guarantee regarding the order in which responses are received
- ☑ Avoid application design relying on ordering of requests, favoring a functional programming approach



# Potential Hazards - Development

- ✘ There are limitations on the number of active XMLHttpRequests
- ✘ And, they are processed in no particular order
- ☑ Avoid application design relying on ordering of requests, favoring a functional programming approach



# Optimization Techniques and Tips

- When changing the content of a node in the DOM tree, build the string entirely then add to the DOM
  - DOM operations are costly
- Don't neglect careful site design, and judicious use of Ajax techniques
- Carefully design the request/response message content, especially XML
  - JSON (Javascript objects)
  - Text



# Optimization Techniques and Tips

- Ajax itself is an optimization, with careful design
  - Small message sizes
  - Caching
- Recommend creating an XMLHttpRequest as needed, rather than reusing
- Use batching techniques
  - Explicit click submission to reduce trips, improve UI intuitiveness, and provide clear Ajax functional alternative paths
  - Queue requests, and send on timer or in batch



# Optimization Techniques and Tips

- If possible, pre-caching or guesstimates could be made to provide predictive behavior
- Some use client-side XSLT for rendering of response results rather than DOM
  - This can make for more maintainable code
  - Performance can be faster
- Utilize standards where possible to avoid excessive future refactoring and improve longevity
  - DOM
  - CSS



# Optimization Techniques and Tips

- Be mindful of the distinction between web applications and web sites
- Clearly determine requirements, such as target browser and accessibility, as soon as possible



## Design Patterns

- Ajax is being used in many creative ways
  - As-you-type suggestions, corrections, validation, search result counts, etc.
  - Refinement of search results
  - Drag and drop items within a page
  - Tree view drilldowns with as-needed content retrieval
  - Predictive pre-fetching of content during idle cycles
  - Morphing display in response to user actions
  - Selectively add and remove items from a page
  - Browser-side polling of the server for results on a timer, periodic client-initiated content refresh



# Arguments Against Ajax

- It's just DHTML and remote scripting with a new name
  - True. Is that really a reason to avoid using it?
- Lots of requests means lots of traffic over the wire.
  - Careful design of the requests should actually reduce the amount of data going over the wire, not increase it.
  - In other words, there may be more requests, but there should be significantly less data and fewer repeated page loads.



# Arguments Against Ajax

- Javascript is not an enterprise language
  - Perhaps not. But it is what we have for standardized client side web development.
  - Toolkits and libraries provide more beefy, more reliable and more standardized functionality
  - Perhaps Ruby will overtake Javascript, but it hasn't yet



# Arguments Against Ajax

- Design, design design... If careful design is important for Ajax, why not put the effort into a traditional web application approach?
  - For graceful degradation and accessibility, a site should handle both sides.
  - Ajax is not meant to replace web applications, but enhance and enrich the user experience
  - There is no substitute for careful design in this business regardless of technology, language, platform, environment, etc.



# Arguments Against Ajax

- Ajax is not cool, it makes pages behave in unexpected ways.
  - A well designed Ajax site should be more intuitive, not less.
  - Poor UI design is not unique to, nor created by Ajax



# Additional Food For Thought

- Many potential hazards and arguments against, apply to web applications in general and are not unique to Ajax
- With each passing year web users get more savvy, and techie gadgets are pushing developers to be even more innovative
- Web 2.0 is hot right now, redefining user interaction with the web application is timely
- It's refreshing to see an old thing made new again, and it's good to see remote scripting finally come into its own



# Looking Ahead, Looking Back

- Tiny Javascript applications, bookmarklets and widgets are gaining attention
- Remember Active Desktop?
- RSS feeds and various forms of content delivery is very popular
- Remember PointCast?
- Blogosphere
- BBS and newsgroups...



# Looking Ahead, Looking Back

- Web 2.0
  - Lots of user input and stranger collaboration
  - Useful, small bits of functionality
  - Service based view of web
  - Free-form, ad-hoc, and focused on user experience
  - Focus on refinement and revisiting of old ideas with a new twist



# Conclusion

- Ajax is really a new name for the old ideas of DHTML and remote scripting
- Ajax applications may be written in a relatively straightforward fashion “in the raw”
- There are a number of Ajax toolkits available to help simplify development
- Web application “gotcha’s” apply to Ajax applications, along with the additional quirks of back button, bookmarking/history issues, and cross-domain limitations



## Resources

- <http://swik.net/Ajax>: Very good Ajax resources, collaborative "mistakes" page here
- <http://ajaxpatterns.org/>: Wealth of resources for Ajax
- <http://www.adaptivepath.com/publications/essays/archives/000385.php>: The original article
- <http://www.ajaxian.com/>: a great resource
- <http://www.xml.com/>: O'Reilly site with excellent articles on XML
- <http://www.sourcelabs.com/blogs/ajb/>: Alex Bosworth's blog
- <http://www.w3.org>: Website for the W3C



## Resources

- <http://www.webaim.org>: Web Accessibility
- <http://www.sitepoint.com/blogs/2006/03/15/do-you-know-your-character-encodings/>: Excellent blog on character sets
- <http://www.xml.com/pub/a/2005/11/09/fixing-ajax-XMLHttpRequest-considered-harmful.html>: Cross-domain issue and sample solutions
- <http://www.oreillyn.net.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html>: Great article on Web 2.0
- <http://atlas.asp.net/docs/Default.aspx>: Atlas documentation
- [http://slayeroffice.com/articles/innerHTML\\_alternatives/#6a](http://slayeroffice.com/articles/innerHTML_alternatives/#6a): Article on how to rid yourself of the innerHTML habit.



# Resources

- As a follow-up, you will receive a zip file containing demo code, resource links page, and useful toolkit code and libraries
  
- Happy Ajax'ing!