

AJAX 101

Asynchronous JavaScript and XML

Southeast Ohio Macromedia User Group



Who's The Rob?

- Name: Rob Gonda
- Codename: The Rob
- Location: Aqua-base One
- Employer: iChameleon Group (CTO)
- Journal: Ajax Developers Journal (E-in-C)
- Access: Group Level 7

TOC

- Definition
- Examples
- History
- What, why, how Ajax
- Ajax Vs. Classic Web
- Potential Problems, barriers of entry
- XMLHttpRequest
- Cross-Browser compatibility
- Libraries, frameworks
- Security
- Tips and tricks
- Debugging
- Do's and Don'ts

What is AJAX?

- Asynchronous JavaScript + XML
- Interacting with the server without refreshing the page
- Rich Internet Applications with JavaScript
- A geek marketing term
- Venture Capital Magnet

AJAX Sites

- Google Mail, Maps, Reader, Suggest, Personalized Homepage, Writely
- 37signal's Basecamp / Backpack
- Backbase
- Zimbra Collaboration Suite
- Bindows
- openlaszlo.org

What is AJAX? (tech view)

- Asynchronous data retrieval using **XMLHttpRequest**
- Data interchange and manipulation using *_XML_* (or not)
- Dynamic display and interaction using Document Object Model (DOM)
- JavaScript binding everything together
- Flash Remoting for JavaScript? Not really, but close

History – Ajax is not new

- Hidden Frames since mid-90's
- XHR and ActiveX since IE5+, Mozilla 1.0+, Safari 1.2+, and Opera 7.6+
- February 18, 2005: Jesse James Garret wrote the article that coined the AJAX acronym.

Why is it popular?

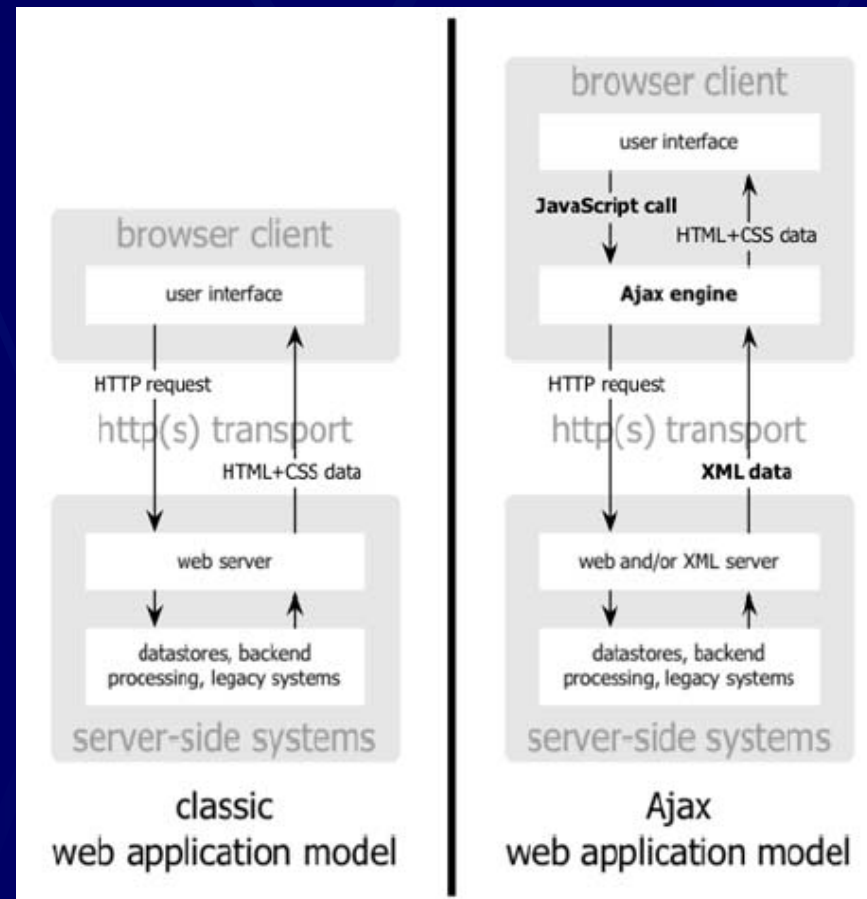
- Google helped popularize, and legitimize it with Gmail and Google Maps
- Increase Usability of Web Applications
- Rich Internet Applications without Flash
- Save Bandwidth
- Download only data you need
- Faster interfaces and better user experience

How can Ajax help me?

- Form Validation
 - move business rule validation to the server while still enforcing them in real time.
 - Check usernames, passwords, emails, etc.
- “pulling data” - update scores, stock quotes, weather, etc.
- Rating, voting
- Input suggest: i.e. Google suggest
- Edit in place
- Active Search
- Filtering large record sets of data
- Pre-fetch data sets to improve speed
- Chat: Ajax Vs. Comet Vs. XML Sockets

Classic Vs. Ajax Web

No more single
request / response
restrictions



Potential Problems

- Breaks back button support
- URL's don't change as state changes
- SEO / Search Engine Indexing
- Cross Browser Issues can be a pain
- Cross-domain requests (SOA, WS)
- Viewable Source Code
- Client side business logic
- Server Load if not properly coded
- Concurrency

Real Barriers to Entry

- Not technology
- Design Elements
- Application design / architecture is not procedural or sequential
- Think beyond the click
- Do not expose your code

XMLHttpRequest (XHR)

- A JavaScript Class that lets you make asynchronous HTTP requests from JavaScript
- Allows to kick off HTTP requests in the background
- A call back JavaScript function is invoked at each state of the HTTP request and response

XMLHttpRequest Methods

- **open** – setup a request
- **send** – sends the request
- **abort** – aborts any request
- **getAllResponseHeaders** – Returns a Map of headers
- **getResponseHeader** – Returns a header value
- **setRequestHeader** – Set header

XMLHttpRequest Properties

- **onreadystatechange** - call back function for state changes
- **readyState** - the current state of the HTTP call
 - 0 = uninitialized, 1 = loading, 2 = loaded, 3 = interactive, 4 = complete
- **responseText** - the text result of the request
- **responseXML** - XML DOM of response
- **status** - HTTP status code of the response
- **statusText** - HTTP status text

Cross-browser: IE

- `new ActiveXObject("Microsoft.XMLHTTP");`
- Methods and properties are identical
- You can't totally blame them because they invented it
- Native XMLHttpRequest support should be in IE7, although you still need to support older browsers

Cross Browser AJAX 1/3

```
var req;
function loadXMLDoc(url) {
    req = false;
    // branch for native XMLHttpRequest object

    if(window.XMLHttpRequest) {
        try {
            req = new XMLHttpRequest();
        } catch(e) {
            req = false;
        }
    }
}
```

Cross Browser AJAX 2/3

```
// branch for IE/Windows ActiveX version
} else if(window.ActiveXObject) {
  try {
    req = new ActiveXObject("Msxml2.XMLHTTP");
  } catch(e) {
    try {
      req = new ActiveXObject("Microsoft.XMLHTTP");
    } catch(e) {
      req = false;
    }
  }
}
```

Cross Browser AJAX 3/3

```
if(req) {  
    // set up callback handler function  
    req.onreadystatechange = processReqChange;  
  
    // set up URL and verb method  
    // the last boolean indicates asynchronous call  
    req.open("GET", url, true);  
  
    // trigger call.  
    req.send(null);  
}  
}
```

AJAX Libraries

- Many people opt for AJAX libraries.
- Provides many advantages
 - Sync / Async
 - Serialization
 - Multithread
 - Error handling
 - Logging
 - Security, encryption, obfuscation
- Disadvantages : NONE!
 - Weight
 - Loads more than you need

JavaScript AJAX Libraries

● Libraries

- Prototype (the most popular, used by RoR)
- Backbase
- Dojo (used by Open Ajax)
- DWR (integrate with Java Struts and EJB)

● Toolkits

- Microsoft Altas (commercial)
- Open Ajax (IBM, Zimbra, Dojo | open source)
- Open Lazlo (Flash and Ajax)
- Tibco

JavaScript UI Libraries

● Widgets

- ActiveWidgets
- Backbase
- SmartClients
- Dojo

● Low Level Effects

- Script.aculo.us (req. prototype, used by RoR)
- Rico (req. prototype, simple)

ColdFusion AJAX Libraries

- ajaxCFC
 - OOP, CF extends objects, Design Patterns
 - Built-in error handling, security, debugging
 - Capability to integrate with Model Glue (soon Mach II)
- CFAjax
 - First Ajax CF Framework
- JSMX
 - Client side library only
- Simple Remote Scripting (SRS)
 - Uses iframes

Security Concerns (I)

- JavaScript applications are easily decoded and reengineered
- On-demand (server side) loading will not help if you load your entire application and business logic in the client's memory
- Obfuscation makes it more difficult, but can also generate bugs

Security Concerns (II)

- XMLHttpRequest is nothing more than a normal form submission
 - Authentication elements
 - Session cookies
 - Blank Referrer by default ← You should manually set this header in your API
- Get / Post Verbs are sent in plain text
 - Consider encrypting requests and obfuscating responses
- Developers forget to send sensitive data over SSL

Security Rules

- Don't trust user input
- Do not trust client side validation
- Do not trust server side AJAX validation
 - Will improve user experience
 - Will not reduce code, only increase it.
 - You still need to re-validate in the server side
- Do not assume every Ajax request is real
- Keep you business logic in the server
- No framework is yet encrypting transmitted data

Do not expose business logic

- Most important aspect for Enterprise Applications
- Ajax uses JavaScript, but it does not have to reside in the client
- Use remote calls only as a transport layer
- Transport state and instructions, not raw data
- MVC
 - Model must remain on the sever
 - View or presentation layer is managed with DOM
 - Controller layer simply requests commands and dynamically evaluates them

Cross-site Myth

- Impediment to communicate with other domains is good. Cross-domain requests are a security hole
- There is no real need for it, Web Services should be consumed by server side scripting
- If really needed, use iframes technique instead

Weight and Performance

● Minification Vs. Obfuscation

- A minifier removes the comments and unnecessary white space from a program.
- An obfuscator also minifies, but it will also make modifications to the program
- Any transformation carries the risk of introducing a bug.
- Easier to minify dynamically generated code than obfuscate it

● GZIP

- 15% of original file. Makes Minification and Obfuscation file size modification almost irrelevant.

Debugging

- `window.onerror = function(message, url, lineNumber) {
 alert("Error: " + message + ". At " + url + ", Line " + lineNumber);
}`
- Use `onerror` to trigger a simple Ajax call that never ever changes to log the error in the server side (text file, xml, database)
- HTTP Traffic Sniffers
 - Live HTTP Headers (Firefox)
 - FireBug (Firefox)
 - Fiddler (IE)
 - Ethereal (Advanced, multipurpose)

Do's and Don'ts

● Do's

- Use Ajax!
- Plan your application, more thought than standard web sites
- Keep your business logic in the server
- GZIP data transmission
- Provide progress indicator and visual feedback. The user is used to page refreshes
- Design for error, not only for success: The What if Factor

Do's and Don'ts

● Don't's

- **Use Ajax for the sake of Ajax.**
- Assume DOM or JavaScript calls are universal. Every browser is different, furthermore in different OS.
- Modify elements unexpectedly. Users are used to trigger changes.
- Break the back button and direct links
- Fetch important content that has to be indexed by search engines
- Perform some server action without a proper visual representation
- Overuse XML. XML is heavy and JavaScript objects or JSON improve performance.

ajaxCFC simple echo

```
<script type='text/javascript'>
  _ajaxConfig = { '_cfscriptLocation': 'echoTest.cfc', '_jsscriptFolder': '../js' };
</script>
<script type='text/javascript' src='../js/ajax.js'></script>
<script type="text/javascript">
  function doEcho() {
    DWREngine._execute(_ajaxConfig._cfscriptLocation, null, 'echo', $('echoInput'), doEchoResult);
  }

  function doEchoResult (r) {
    $('echoScreen').innerHTML = $('echoScreen').innerHTML + '<BR>' + r;
  }
</script>
<input type="Text" id="echoInput">
<input type="submit" value="enter" onClick='doEcho(); '>
<div id="echoScreen" style="height: 100px;overflow: scroll;"></div>
```

setup

Ajax call

Callback function

Input field

Echo _screen_

ajaxCFC Server Side

```
<cfcomponent extends="ajax">  
  <cffunction name="echo" output="no" access="private">  
    <cfargument name="args" required="Yes" type="array">  
    <cfreturn arguments.args[1] />  
  </cffunction>  
</cfcomponent>
```

Thank You

- Questions / Comments?
- Blog: <http://www.robgonda.com>
- email: rob@robgonda.com