Some Issues on Ajax Invocation

I. Introduction

AJAX is a set of technologies that together a website to be -or appear to be- highly responsive. This is achievable due to the following natures of AJAX[1]:

1. Partial refresh: when an interaction event fires, the server processes the information and returns a limited response specific to the data it receives. Hence, server does not send an entire page, as is the case for conventional web applications.
2. Asynchronous: after sending data to the server, the client can continue processing while the server does its processing in the background. This means that a user can continue interacting with the client without noticing a lag in response.

The following figures will show comparison between traditional synchronous model against AJAX-enabled asynchronous model[2]:

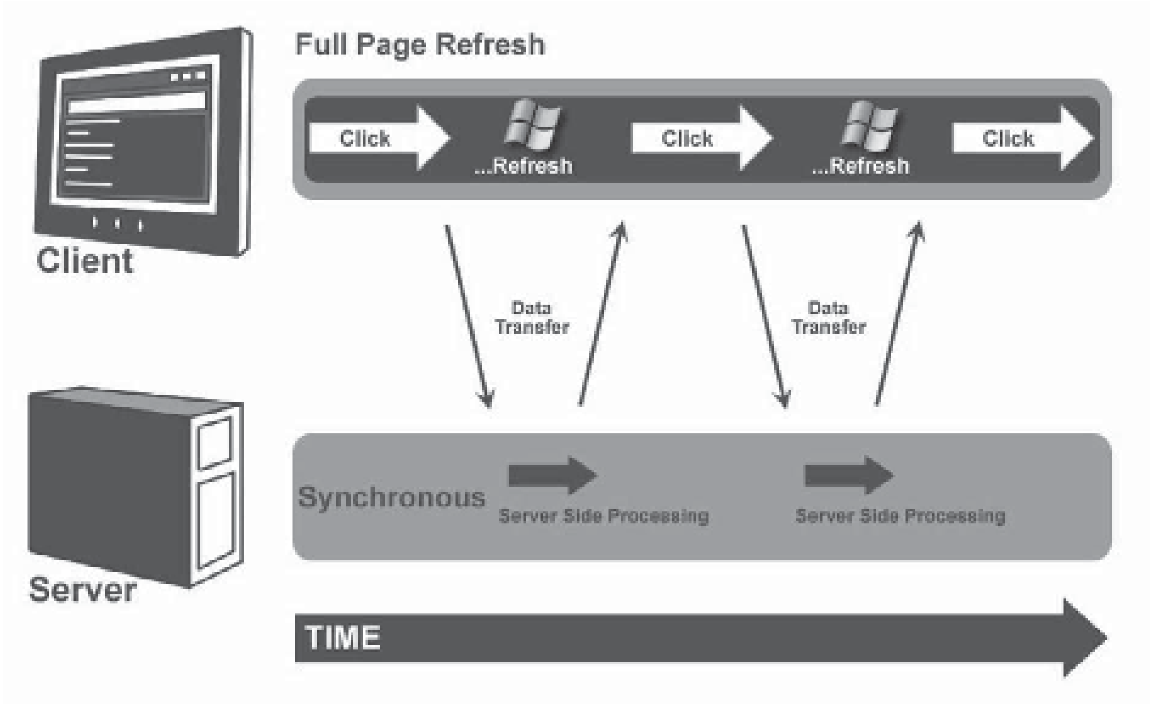

Figure 1: Traditional synchronous model

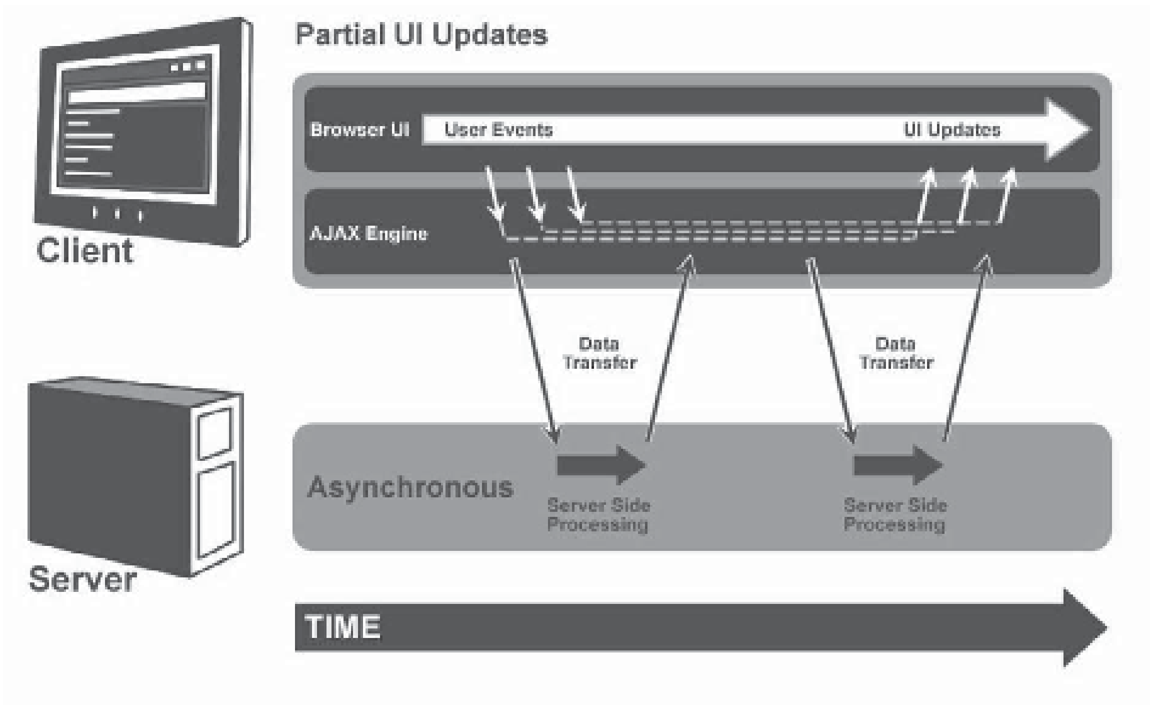

Figure 2: AJAX-enabled asynchronous internet model

Besides AJAX-implementation depicted in Figure 2, there is also another way of implementing AJAX called named AJAX Push (or Comet). The difference between Comet and standard Ajax is Ajax Push allows creation of event-driven web applications which are hosted in the browser[3].

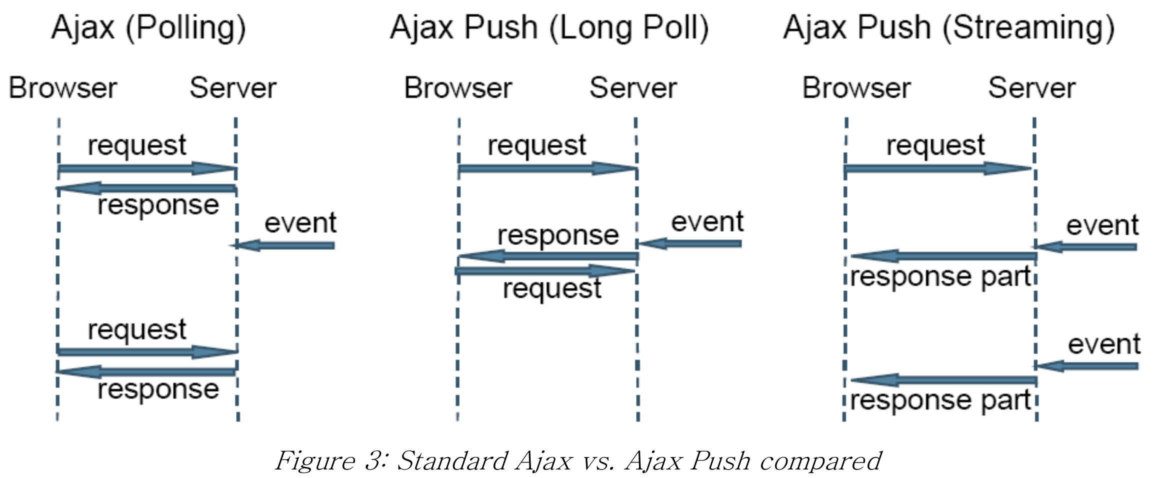

Figure 3: Standard Ajax vs. Ajax Push compared

AJAX invocation usually needs the following components[1,4]:

1. CSS: in an Ajax application, the styling of an application may be modified interactively through CSS

2. Javascript, a scripting language. One element of Javascript that is key to AJAX is XMLHttpRequest (XHR)
3. DOM (Document Object Model). DOM represents the structure of a web page as a programmable tree structure. Scripting the DOM allows an AJAX application to modify the UI on the fly, effectively redrawing parts of a page.
4. XML, the format for sending data from web server to the client. However, other formats can also be used, like HTML, JSON, or plain text.

The above components can be rephrased in diagram below which shows HTTP request comparison between classic web application model and AJAX web application model:

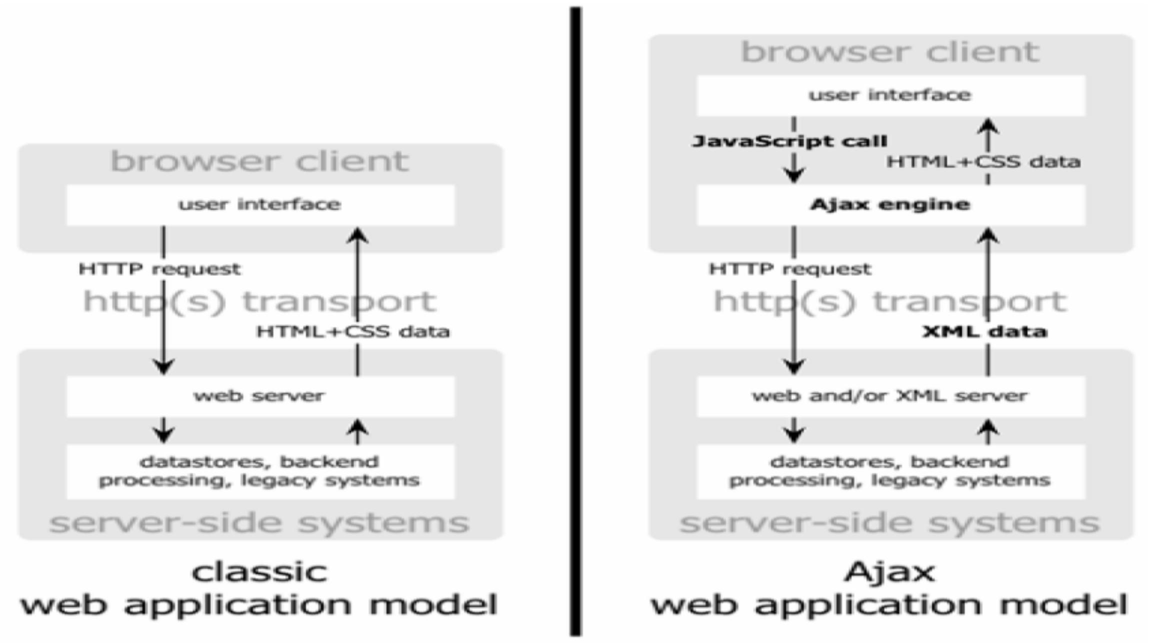

Figure 4: Classic vs. AJAX web application model

II. AJAX Issues

As new technology always brings issues with its inception, similar case also applies to AJAX. I will divide the issues into following categories:

1. Application design issues
2. Performance issues
3. Usability issues
4. Security issues
5. Mobile application issues

II.1 Application Design Issues

With AJAX, user can easily spawn off numerous requests to various web services using REST or SOAP interfaces and even route data from one to another. This will create concurrency issues within the application[5].

The concurrency issues are listed as the following:

1. Internet Explorer limits the number of concurrent requests to particular unique FQDN to 2 connections at a time.
   Possible solutions: registry tweaks, use different virtual domain names for the same server
2. To have multiple requests, one must ensure the the XHR object is unique particularly if using a global variable.
3. One must use closures in order to guarantee that callback to response object is appropriately passed.
4. As concurrent requests will be running asynchronously, it is impossible to know which will return first. The callback function must take that into account.
   Possible solution: implementing semaphore concept
5. Possible race conditions with session variables.
   Possible solutions: re-architect application, locks

Issue number (5) is perfectly described by the following scenario:

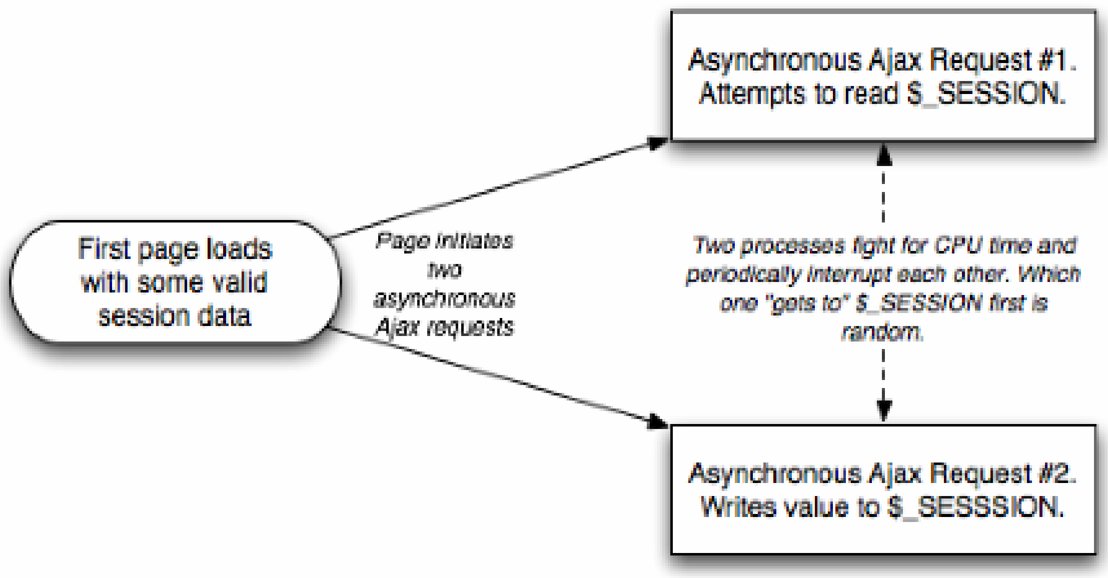
Asynchronous Ajax Request #1. Attempts to read $_SESSION.
First page loads with some valid session data
Page initiates two asynchronous Ajax requests
Two processes fight for CPU time and periodically interrupt each other. Which one "gets to" $_SESSION first is random.
Asynchronous Ajax Request #2. Writes value to $_SESSSION.

Figure 5: Race condition issue in Ajax

II.2 AJAX Performance Issues

An important use of Ajax is to enable incremental updates to the web page. This is doable because of the asynchronous request approach. Depending on the type of page and the amount of data needed to be updated, it should be possible to reduce the amount of network traffic generated.

However, while this assumption might be true, there are problems with performance as noted below[6]:

1. Excessive server requests

Problems:
- one single action can generate a flood of requests
- however, one can not determine cause of requests without page analysis
- one missing timeout can produce a huge backlog

2. Javascript problems
   Problems:
   - not as fast as compiled languages
   - processing large data sets is hard
   - hard to debug nested DOM/CSS/JS
3. Server load
   Problems:
   - Initial download of engine (static content)
   - bursts of activity (change of content)
   - frequent requests and polling use network, CPU, sockets, and threads
4. Service fragility
   Problems:
   - each service is a possible point of failure (outages, configuration problems, changes to service)
5. Integration server problems
   Problems:
   - slow database query
   - too many queries
   - bottlenecks from aggregating many services
   - contention for locks in server because lots of threads for one session
   - outages or misconfiguration

II.3 AJAX Usability Issues

These issues relate with user experience when browsing an AJAX-enabled web page. The issues are listed as follows[7]:

1. Back-button issue
   →Ajax-based application needs special technique to simulate "Go Back" process
2. User can not bookmark pages
   →Currently it's not possible to bookmark the url and content of Ajax-enabled page
3. Server errors get lost
   →Unexpected server responses during Ajax operations which is untrapped can result in the appearance of a "hung" application
4. Longer initial page load time
   →Ajax will likely increase initial page load time since it must load client libraries to handle all of the Ajax operations.
5. Search index problems
   →Data loaded after initial page load won't be crawled by search engine

III.4 AJAX Security Issues

Below is the summary of security issues in AJAX-based applications[2,4,8,9]:

1. Client-side data validation and verification
   An improperly formatted request sent to the server will cause application to perform unnecessary processing that may cause the application to enter into invalid or unpredictable states or return exceptions and errors. With AJAX implementation, extra validation and verification needs to be done on the server (initial and final data submitted)
2. Cross-site scripting and script injection
   Ajax amplifies Cross Site Scripting (XSS) vulnerability. With Ajax, XSS can make malicious requests with a user's credentials without refreshing the web page.
   Example of AJAX XSS vulnerability: Samy worm[2].
   Another type of script injection is SQL injection through unsafe AJAX application.
3. Session tracking
   Two problems related with session tracking is 1) adding session information to a request which is not under public service domain and 2) corrupted session data due to the asynchronous nature of AJAX.
4. Client-server model
   With AJAX the program is now distributed across two computers (client and server) instead of on the server only as in the classic approach. With this model, writing correct programs is more difficult hence increasing the risk of programming errors and also security issues.

IV.5 AJAX Issues in Mobile Application

AJAX is also considered to be implemented in mobile world. However, there are still constraints and limitations in mobile device hardware and wireless networks which leads to barrier for AJAX implementation[10,11]. The list can be seen below:

1. Performance
   Ajax requires optimized performance for running complicated Javascript code on mobile devices.
2. Memory consumption
   Most mobile devices are shipped with limited memory size while AJAX application usually takes significant memory resource.
3. Limited key and screen input
   Many mobile devices have limited key inputs which create significant difficulties when developing rich web applications for mobile devices. Smaller screen size may also bring difficulties for UI designers and mobile device user.
4. Battery life
   Data transmission tends to be draining battery power quite heavily. Javascript XML/DOM/JSON operations mean heavy CPU load which not only results in slow performance but also in battery drainage.

References

[1] Ort, E., Basler, M. *AJAX Design Strategies*. http://java.sun.com/developer/technicalArticles/J2EE/AJAX/DesignStrategies/design-strategies.pdf

[2] Ritchie, Paul. *The Security Risks of AJAX/Web 2.0 Applications*. http://www.infosecurity-magazine.com/research/Sep07_Ajax.pdf

[3] Arcand, Jeanfrancois. *Asynchronous Ajax (aka Comet) Using the Grizzly Comet Framework*. http://weblogs.java.net/blog/jfarcand/archive/OnTheRoadToJavaPolis2007_Comet.pdf

[4] Forum Systems Inc. *Ajax Application Security and Performance Considerations*. http://forumsys.com/papers/ForumSystems_AJAXwp-spring2006.pdf

[5] Powel, Thomas A. Programming Language Principles and Paradigms. Lecture Notes Spring 2007: Chapter 13 Concurrency + AJAX Discussion. University of California San Diego. http://classes.pint.com/cse130/lecture20/big.pdf

[6] Bodkin, Rod. AJAX Performance and Monitoring. Colorado Software Summit 2006. http://www.softwaresummit.com/2006/speakers/BodkinAJAXPerformanceAndMonitoring.pdf

[7] The Ajax Papers – Part III: Why, When, and What. http://www.telerik.com/documents/AJAX%20Page/Ajax-Part3.pdf

[8] Sonntag, M. *Ajax Security in Groupware*. Proceedings of the 32nd EUROMICRO Conference on Software Engineering and Advanced Applications (EUROMICRO-SEAA'06). http://ieeexplore.ieee.org/iel5/11118/35633/01690173.pdf?tp=&isnumber=&arnumber=1690173

[9] Hoffman, Billy. Ajax Security Dangers. http://www.spidynamic.com/assets/documents/AJAXdangers.pdf

[10] Harakawa, Takuyu. Embedded Ajax – Web 2.0 Optimized for Mobile Devices. http://www.w3.org/2007/06/mobile-ajax/papers/access.sasaki.PositionPaper_EmbeddedAjax_ACCESS.pdf

[11] Jeon, J., Lee, S. Position Paper: Toward a Mobile Rich Web Application – Mobile AJAX and Mobile Web 2.0. http://www.w3.org/2007/06/mobile-ajax/papers/etri.jeon.MobileAJAX-PositionPaper-r5.pdf