| White Paper Submission Template | White Paper Type (select type from dropdown): |
|---|---|

Enter text in the grey areas. After you complete your submission, save your document with a name for your white paper and upload the document to the Product Engineering White Paper Submission library. Your submission should not exceed 4 pages.


Contact Information

Name: Darrell Pratt Email: darrell.pratt@nielsen.com Phone: 312.285.0616 Location: Schaumburg


Have you submitted an invention disclosure? No

If yes, enter the Anaqua Invention Disclosure Record Number

## Executive Summary


With the move away from Adobe Flex, JavaScript/AJAX based applications are quickly becoming the standard for new projects at Nielsen. These applications can provide the client with rich experiences on par with their Flex counterparts and are generally lighter and more broadly supported across mobile devices and modern browsers. Many libraries are available to improve cross-browser support and to speed time to market by providing frameworks to build upon. Nielsen needs to standardize on a subset of these libraries to build a set of best practices and work with generally accepted MVC design patterns.


## Problem Statement

Adobe Flex provides a plugin environment for applications to run within and can generally be counted on to work identically across different browsers. In addition to this cross browser capability, the Flex API is a complete programming environment and Adobe APIs such as Cairngorm and Parsley are generally accepted frameworks that can be used for standardizing coding principals across development teams. However, Flex is not supported on a large portion of the current mobile device market and looks to be removed in future versions of Android. HTML5 and JavaScript can provide Nielsen with an equivalent set of capabilities as Adobe Flex, but care needs to be taken now to ensure a standardized approach to building applications with these technologies at Nielsen.

Many JavaScript libraries are on the market now and each provide separate methodologies for application architectures and varying levels of cross browser support. Short of building our own Nielsen standard JavaScript library, we should attempt to standardize on a subset of these libraries which can provide various advantages for each type of application that is being built. In general there is no one solution to a web application, but a set of 3-4 libraries could solve most browser issues and provide a good set of tools for most of Nielsen's web application needs.

This paper will list a set of libraries that have been used within the GPS group and their various pros and cons. Lessons learned regarding testing, continuous integration, deployment and optimization will be discussed as well.

## Previous Options

Adobe Flex has been used within Nielsen to construct rich user interfaces. In addition to Flex, Tibco General Interfaces has also been used to create graphical applications using JavaScript and HTML. Both of these solutions are nearing their supported lifetime from their vendors and Nielsen must now explore other options for building such applications.

Adobe Flex provides a plugin based approach to developing applications where ActionScript is the language used to develop the various components of the application. A designer application is available to assist with the creation of the user interface but generally more complex applications are hand coded by the developer.

Tibco GI provides a graphical interface to generate the application with adapters to tie into various data sources for the data to present in the application. The code generated is not generally regarded as clean or commonly accepted style of coding JavaScript however.

Both of the above applications have been either end-of-lived by their manufacturer or are in the process of being moved to an open source model. At this point, moving toward more broadly accepted JavaScript/HTML5 solutions is in the best interest of Nielsen.

## Proposed Solution

Two popular JavaScript frameworks can be used to replicate the functionality of a typical Flex based application at Nielsen. One solution, jQuery, is an open source framework that is very popular due to its capacity to be extended with plugins and has a large community of support. jQuery by itself is primarily a DOM framework, meaning that it is used to manipulate the document object model of the application and also provides various utility methods for dealing with AJAX requests to a server side application. The second framework in use at Nielsen is the Sencha ExtJS framework. This option provides both DOM and AJAX utilities, but also gives an entire UI and MVC modules. Sencha is licensed as open source with the MIT license, but is also commercially supported by Sencha Inc.

Both frameworks are pure JavaScript and thus run in the browser client natively which provides measurable performance gains over a plugin framework like Adobe Flex.

### jQuery

jQuery consist of a small library that aids the developer in dealing with the document object model of the application, animations, event handling, and Ajax interactions with the server. A rich plugin framework is also available that has resulted in a large 3rd party environment of many other plugins which provide functionality for all number of applications. The library does not provide any UI options as some other frameworks do, but many plugins have been written, such as jQuery UI[1], that do provide the standard set of user interface elements to the developer. These include tabs, modal windows, tree selectors, grid views and many others. Typically an MVC framework is added to the standard jQuery library to provide the developer with framework for creating complex applications. Backbone[2] is one such library that is in

[1] http://jqueryui.com/

[2] http://documentcloud.github.com/backbone/

wide use and provides a standard methodology for adding models, views and controllers to the JavaScript application to mimic what one would create in a Flex based solution.

**Sencha ExtJS**

The ExtJS[3] framework gives the developer a complete library of utility classes, UI elements and a rich MVC framework for creating applications. This framework is a complete application environment which provides not only the libraries, but a build framework with JavaScript builder support through to a development environment for laying out applications and generating the scaffolding code for new applications.

ExtJS provides an abstraction layer above the HTML5 specification. Many classes within the framework provide a facade over the newer features and can be coded to and provide fallbacks to legacy browsers which do not support these features. An example of this is the local storage support for offline applications that is present in WebKit browsers. ExtJS gives the developer support to use the storage mechanism if it exists, and can fallback to various other methods for older browsers.

Another benefit to the ExtJS framework is the sibling product Sencha Touch. This product shares a common code base and gives the application author a full set of mobile browser ready classes and widgets to build mobile applications. Sencha also provides tools to package HTML based applications into native application bundles for both Android and iOS.

## Additional Detail and Requirements

Both frameworks would require development teams to have a good understanding of JavaScript to build applications that would be on par with Flex solutions. JavaScript has grown from a solution to create image roll overs into a full fledged programming environment that is based on prototypical inheritance with rich closure support. Although Actionscript from Adobe is based upon the same ECMA standard, JavaScript has advanced a great deal and requires that the developer become familiar with a different methodology of writing code that is asynchronous in nature. This can be challenge for some, but there are good resources online and various training options on the market to bring development teams up to speed. Sencha offers a bootcamp class for both ExtJS an Touch frameworks that has been attended by Nielsen employees with great success.

## Risks, Issues, and Dependencies

The primary risk of moving to JavaScript based applications for Nielsen is the varying levels of support across the current browser environment. With Flex, this is mitigated with the Flash plugin, but JavaScript relies completely on the browser for its execution environment and thus can yield varying results across different instances of browser makes and versions. This can be mitigated with the use of shims and polyfill[4] frameworks that can backfill support for modern objects and access methods on older browsers. As long as the frameworks chosen have good backward compatibility which both ExtJS and jQuery do then this risk is generally diminished.

---

[3] http://www.sencha.com

[4] https://github.com/Modernizr/Modernizr/wiki/HTML5-Cross-Browser-Polyfills

## Benefits

The primary benefit of moving to JavaScript as our framework for creating browser based applications is that the use of JavaScript provides a very light weight and future proof application environment. These frameworks both provide support for mobile browsers whereas Flex is a dead end in those environments. Applications built using either framework provide a codebase that can be re-used across desktop and mobile browsers and this is a significant cost savings going forward.

## Implementation

There are two programs already moving from Flex to ExtJS for client facing applications within Nielsen. Both of these applications are scheduled to be released in July 2012 and can both be used as showcases for best practices when using this framework to build rich web applications. The libraries and build methodologies of each should be published on an internal iShare site so as to allow other groups to learn from these programs and to re-use any code. In addition a series of brown bag sessions can be conducted across engineering teams to share the knowledge that has been gained by these groups.

A system of shared modules similar to Github[5] should also be considered to share common code across departments. Teams could contribute common code libraries and build methodologies to the Nielsen company as a whole.

## Summary

Moving from Adobe Flex to standards based JavaScript frameworks for rich internet applications will not only generate lighter weight and more performant applications, the applications will also be available on most mobile browsers and allow for a common code base between both mobile and desktop applications. JavaScript as language is extremely advanced and comes with a vibrant community of developers that are pushing the language to provide greater functionality across many domains.

[5] http://www.github.com