International Journal of Advances in Computer Science & Its Applications– IJCSIA
Volume 5: Issue 1 [ISSN : 2250-3765]
Publication Date : 30 April, 2015

Rich Internet Application (RIA) New Dimension With HTML5, CSS3 and Javascript Technology.

Shamsul Anuar Abdul Wahid, Zee Kum Khoon

***Abstract*— Early generation of Rich Internet Application (RIA) technologies fulfill the requirement of RIA through plug-in solutions, and browser specific extension provides attractive solution of running native like application as a client to web backend, but fragmentation of the technologies and nonstandard implementation have hindered their wide adoption.**

Introduction of HTML5 and CSS3 implementation in all the major browsers recently has changed the RIA landscape. Now we have a technology stack that is controlled by a standard body, and widely implemented in mainstream browsers as a base to create RIA applications. Other than the control and standardization, this technology stacks are also exceed the original definition of RIA.

As a result, the market sees a flood of browser based RIA that doing things previously can only be done by native applications.

This paper describes our own experience in implementing a RIA application based on HTML5, CSS3 and Javascript and highlight how these technologies have exceeded the old RIA definition.

***Keywords*—RIA, HTML5, Web Technology**

I. Introduction

Rich Internet Application (RIA) is a category of web software applications that leverage on the web technology to look and behave like native applications. This category of web applications need to fulfill four basic requirements to be a RIA[5] :

a) Instead of relying on caching mechanism such as cookies, RIA can use client machine native storage.

b) Web application presents different function through different pages. Generally, the browser is refreshed to present the Graphical User Interface (GUI) sequences that represent the user workflow and navigation. In RIA, just like in native application, the single page application (SPA) is used. RIA runs in one main application window, and user functions are presented through sub application windows within the main window. Only specific sub windows is refresh at a time.

Shamsul Anuar Abdul Wahid /Department of ICT
MIMOS Berhad
Malaysia

Zee Kum Khoon / Department of ICT
MIMOS Berhad
Malaysia

c) Improved presentation layer includes page layout and user interaction and behavior.

d) Support both synchronous and asynchronous communication methods.

In this paper we analyze the impact of new stack of technologies: HTML5, CSS3 and Javascript on RIA development. We benchmarked the improvement compared to one of the well-known Web 2.0 RIA, Sun's Lively Kernel.

This paper starts with the history of previous RIA technologies, before it dives to the HTML5, CSS3 and Javascript advancement today. Then it go through our RIA project and lesson learned from the project. We touch a bit on new upcoming technologies that we believe going to change the RIA landscape. At the end we close this paper with our conclusion and where we think the RIA development is heading.

II. RIA Technologies and Architectures

The goal to enable RIA application through web has been there since the first generation of HTML static websites deployed. Various technologies have been created to achieve this goal. In this section we go through some of the major technologies which have been adopted during specific era.

A. Plug In Technologies

The earliest attempt to add interactivity to HTML page was through browser plug ins. The most well-known GUI plugin technologies are Sun Java JavaFX [2], Adobe Flash and Microsoft Silverlight.

Plug In Technologies Vendors
Macromedia Flash (1996),
Adobe Flash (2005),
Microsoft Silverlight (2007)
JavaFX (2008)

Table 1 – Timeline of most well-known GUI plugin technologies

In the pre-HTML5 era, plugins are commonly used to enhance the interactivity of the web application in the browsers at that time. This includes graphics animation, multimedia, hardware and scripting support. Various browser plugins offers different technological advantages as follows.

Adobe Flash, the pioneer in web plugin technology for its success in introducing the web browser flash player to the mass started as early as 1996. Adobe Flash has vector engine

and a range of multimedia codec for rendering its content. It has a Just-in-time development environment and a player that is able to develop, compile and run SWF (Shockwave Flash File format) across various supported devices that is running Android, IOS and also most desktop computer browsers such as Internet Explorer, Firefox, Opera and Google Chrome.

Microsoft Silverlight introduces an integrated development environment and a browser player supported by popular desktop browser and Microsoft Phone. Microsoft Silver has a development environment integrated in Microsoft Visual Studio, which enable Microsoft Visual developer to develop Silverlight control without imposing huge learning curves. The Silver control has an extension .XAP with contents in ZIP format. The XAP is a collection of .NET managed assemblies (.DLL files) and an AppManifest.XAML file defining the entry point (class and assembly) included in the same file. It utilizes the Microsoft Presentation Foundation (WPF) as the core presentation framework embedded in a web page.

JavaFX is another popular web plugin technology that delivers RIA to browser and Java-based desktop/mobile native application. It supports wide range of devices through Java SE for desktop application, Java EE for web browser and Java ME for mobile devices. JavaFX uses Netbean IDE as its primary integrated development environment. In addition, it has several tools as plugin to integrated Adobe Photoshop and Illustrator format directly to JavaFX application. After 4 years from its release, JavaFX introduces scene builder which allow designer to drag and drop UI component to develop application. In future, the integration between JavaFX to JavaSE embedded would means JavaFX to be supported in high-end mobile devices as well.

These few popular browser plugin are widely used by web developers to deliver RIA for many years. Each technology delivers unique technological advantage through proprietary development eco-system and through different scripting languages. The competition has made the Internet filled with variety of RIA and spurned different topic in the field of RIA. These plugin has proved the importance of RIA concept in the web development.

Besides enhancing the multimedia capability of the browser, plugins are also code libraries to support new data format without the need to re-release the browser itself [13]. The development of browser plugin is commonly supported through NPAPI, a cross-browser API for plugin, which is introduced by Netscape back in 1995. Today, most of the well-known browser has plugin API which allow developer to develop and commercialize their plugins in respective browser.

However, security is the main concern for these plugin technologies [9]. As most of these technologies are proprietary, loophole introduced by these plugins are often big concerns to industry leader. For example, Apple has discontinued supporting browser plugin technology in its devices running IOS and MacOS in 2009, but moves to support HTML5 Open Source initiative instead. Canvas was implemented in the HTML5 as substitute to these plug-in technologies running on OS X and IOS.

B. Web 2.0 Application

The term Web 2.0 represents the shift of web consumer habits from expert-driven content to user-driven content. [14]. Web 2.0 application may allow user to consume and produce content for sharing in a virtual community. Social networking sites, blogs, wiki and video sharing sites are few example of Web 2.0 application.

The Web 2.0 era is in the years between 2005 and 2010 [6]. The focus of this era is to build the capability of web technology in delivering Web 2.0 applications on the browser. Some of the most popular web technologies for delivering Web 2.0 applications are Ajax and Web Browser Plugin such as Adobe Flash.

Ajax-based RIA offers more interactive ui, stateless programming model, session maintained in the web servers. While Non-Ajax has all Ajax-based RIA have to offers but added the capability to maintain session in the client-side environment. In addition, Non-Ajax based RIA has client side storage and the capability to better tap into client side hardware such as Graphic Driver, Mic and Speaker.

However, AJAX is widely used for the development of Web 2.0 application. Adobe Flex is the few popular frameworks available for the development of Web 2.0 application. It uses AJAX to fetch the live content from the web server back-end for web application to display without refreshing the web pages.

C. Full Javascript based RIA

Full Javascript based RIA refers to an application that may be developed using any other languages, but render in Javascript. Example of framework that uses this approach are Capucinno, Sencha, SproatCore, Ember and GWT and ExtJS This type of framework may or may not render HTML, CSS and DOM object in the browser. The application developed using this framework is automatic render similarly to native application but runs on the browser.

An example of popular open source Full Javascript based RIA is Capoccino. It consists of its own UI framework, supported language and the capability to emulate native application features such as undo and redo. The advantage of building application using these frameworks often reduce line of codes and shorten the learning curves for Apple native application developer to develop RIA to run on browser.

Today, some of this Full Javascript RIA has been re-innovated with its own programming language such as GWT using DART and Microsoft introduces Typescript. Often, language introduce driven toward each RIA framework that the Technology Leader introduce.

The most popular Javascript based RIA framework available for developer today are often offered in full stack, thus focus and reducing the line of codes for building the application. Aside, supported languages are usually extends the functionality of what Javascript can do.

However, the reception of these frameworks is often limited by the size of its community. As most of this

framework is supported by sizable community, the features are limited. But, enterprise favour these type of framework as it reduces the development lifecycle and yet capable of delivering Full Javascript RIA which can runs on the browser.

III. Web 2.0 RIA Application Reference

Our RIA implementation follow the path almost the same as Sun's Lively Kernel [6], which works on 3 assumptions:

a) World wide web as the target platform

b) Browser is the operating system

c) Javascript as the programming language for web

Lively Kernel is built on top of Web 2.0 technology. Even we created our RIA based on the same assumptions, but the final implementation is difference because we are leveraging on the advancement of the browser support in areas of HTML5 and CSS3. Javascript support has not changed since then.

A. *Web 2.0 Background*

A closer look at Web 2.0. The main idea behind Web 2.0 is the ability for user to generate content. The social media allows user in a virtual community to create, share, exchange information. Others are wikis sites allows user to collaboratively contribute to the content of an encyclopedia. Traditional web application started as web application to consume expert generated content viewed in the browser. However, these platforms evolved to use WOA (web-oriented architecture) which expose the functionally to other application (feeds, RSS, Web Services, mash-up) for integration and platform leverage, thus providing richer application.

The idea for Lively Kernel is to provide a platform for the virtual community to create, share, exchange web component for building RIA web application. The main workflow for creating active content with Lively kernel is to drag objects from the Partsbin to the working environment. It uses morphic-style interface which allow direct manipulation of objects (morph) [15]. A morph has a visual representation that can be picked up and move. These features is driven by SVG+SMIL and closely resemble what Adobe Flash can handle.

B. *Browser Implementation*

Modern Web browser evolved from early thin client architecture. Thin client such as traditional web browser impose some limitation to meet the requirement of modern web application. This section describes how the modern browser implementation overcomes this limitation with the introduction of HTML5.

Browser is built with history which allows user to recall back previous path visited. However, RIA requires state to maintain the user session. As browser is stateless, these handling pose issue to RIA. For example, each time user refresh the screen, it reset back to default state. However, Browser I/O model (reload, stop, back and stop) handling problem[6] is solved using HTML5 History/State APIs. History.js is one of the Javascript library that uses HTML5 History/State APIs in modern browsers to keep the state of the browser while navigating the browser history.

Most of browsers implement an important security concept call same origin policy. This policy prevents access to other methods or properties across pages on different sites. However, this restricts developer to build richer application as it cannot leverage or integration with other platform. [6]Same origin requirement (use proxy), accessing back end data – our solution use of JSON over web service. There are a number of JSON over web service implementation available, such as JSON-RPC.

Web browsers are limited to 2 persistent connections per server. However, different browsers have different persistent connections per server in order to minimize page load time.

Browser Vendor and Version	Concurrent Connection
Firefox 2	2
Firefox 3+:	6
Opera 9.26	4
Opera 12	6
Safari 3	4
Safari 5	6
IE 7	2
IE 8	6
IE 10	8
Chrome	6

Source: http://www.stevesouders.com/blog/2008/03/20/roundup-on-parallel-connections/

Traditional browser does not support offline capability. Web is often associated with the term online. But, modern browser has offline implementation in term of application cache, localStorage, browser DB and online/offline events. Offline capability in RIA enables user to user any web application offline and sync back the data when online. No access to local resources or host platform

Interoperability and compatibility – HTML5 shims are used widely by web developer for enabling their HTML5 application runs on older browser. Basically, shims is a polyfill made in javascript that enabled HTML5 features to be used on older or non-HTML5 browsers. HTML5 canvas is one of the features which are made available in the older browser through the use of shims. Interoperability between different browser vendors is also a challenge for web application developer as different vendors implements HTML5 differently.

In conclusion, these are some of the main browser implementation to overcome the limitation of earlier thin client browser are discussed. These implementation shapes how a RIA works today. Security, performance, on-demand

features, interoperability and compatibility made browser a competitive platform to build the RIA.

C. Back end server

Modern web application architecture is basically consisting of web server and data server. Web Server served web application which consists of the HTML, CSS and Javascript, while the data server exposes API for web application to retrieve and present the data in the view. This architecture can further expanded to include authentication, authorization server and other facilities that a solution might require.

In comparison, Lively Kernel back-end server consists of repository of parts, called Partsbin. Partbin uses WebDAV combined with SVN for storing and versioning components. The combination of WebDAV and SVN allows the persistency of Lively Kernel's components and allows direct access to the components from the web application [10]. Since the component can be directly accessed from the web site itself, the web browser can be the IDE that can be used to build the web pages itself through Lively Kernel Morphic style structures the components The morphic style consist graphic architecture, event handling and green thread scheduler that dictate the shape and the behaviour of the component [10].

IV. HTML5, CSS3 and Javascript RIA

In this section we going to go through the details of the HTML5 based RIA technology stack that we use and how it supports the RIA requirement. The two areas of focus are the level of RIA basic requirement support, and the second is how the architecture of this new RIA is different from previous RIA implementation.

Even there is doubt earlier on the support for HTML5 and its readiness to be the platform to create RIA application [7], but on the other side the industry has started to use the HTML5 stacks / open platform to build RIA [8].

A. RIA Requirement

There are a number of requirement that RIA needs. We map our HTML5 Modern Web Architecture with the RIA model based suggested comparison parameters from JC Preciado [11]. We focus on the model relevant to Web2.0 and interactivity part of RIA.

Visual Continuity - Possibility to avoid screen refreshments and blink experiences

Older web browser uses postback whenever the content needed to be refreshed. Modern HTML5 web application is able to use Ajax call to fetch the content from the server and display in the view. Modern browser also has history and state API built in to manage complex views. It is often found in Single Page Application. The term SPA is coin by Steve Yen in 2005, whereby all necessary web assets, e.g. HTML, CSS and Javascript are retrieved in the single page load. When screen need to be refreshed, rather than post back to the server, a client-side web framework is able to refresh the content through AJAX & JSON call, History and Browser State. Some of the open source tools available are History.JS, Sammy.JS and Jquery.

Multimedia - Possibility to support the representation of graphics, audio, video, streaming and live multimedia

HTML5 web application multimedia support is extended to Canvas, WebRTC and WebSocket . These features provide the multimedia support for graphics, audio, video, streaming and live multimedia to HTML5 Web Application without the need of browser plugins.

Dynamic Data Retrieval Synchronization - Possibility to carry data to/from the server at run time

The possibility to carry data to/from server at run time in application running in web browser is due to local storage and session storage capability built in the modern browser. Such storage is using either IndexedDB (IE, Firefox, Chrome) or WebSQL (Safari, Chrome). There are several JS framework that can synchronization of the data back to the server at run-time. Some of the libraries include Persistence.JS which has client-side library and back-end Node.JS library. Data centric HTML5 web application often uses MVC or MVVM for development.

Personalization - Extensions for internationalization and localization, accessibility, multi-device access, etc.

For personalization in HTML5 web application, the use of cookie in older browser for storage of user data is being replaced by localStorage or sessionStorage in modern browser. Local storage allows persistence storage of the user data until the user cleared the browser cache. Session storage stores user data until user session expired. In term of multi-device access, HTML5 web application uses responsive design and fluid design to allow optimal view in different type of view port. The idea of responsive design is the view is designed to work various viewports through handling the viewport width "breakpoint".

B. Architecture

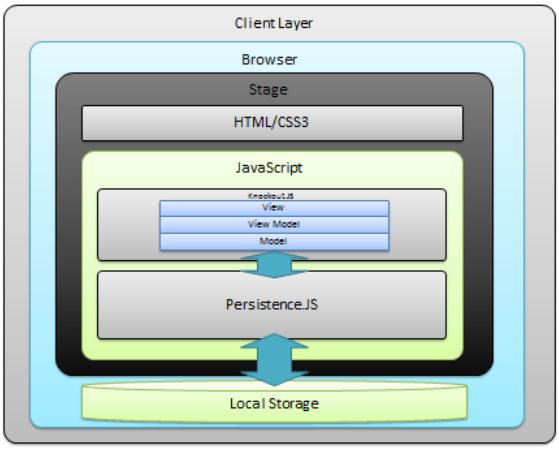

Figure 1- The Web Project platform is separated into 3 block layers: the client layer, the browser and the stage.

Client/Presentation Layer: responsible for presenting information to the user and interpreting user commands.

Browser: responsible to render the stage execution code.

Stage: is a workspace where it controls the page layout, user interaction and behavior of HTML5 and the lifecycle of JavaScript components used to support the presentation layer.

HTML/CSS3: responsible for structuring the page layout and how the information is presented.

Javascript: . The Javascript layer utilizes Javascript tools such as Bootstrap.JS for responsive UI, Knockout.JS for structured web application, Persistence.JS for persistence data store and other JavaScript library to support RIA requirements as mentioned.

Local Storage: responsible for providing a non-volatile data storage in the web browser.

The implementation of structured web application through Model-View-View Model using Knockout.JS and Persistence data storage using Persistence.JS Framework and the interaction between the these components as follows:

- The view requests data from the model to render the UI to the user
- The view-model acts as middle-man to translate the view information to model and vice-versa.
- The model notifies the view-model when the state has been changed.
- The persistence retrieve and updates the data from the web browser local storage to the model or vice versa. Persistence also capable of synchronizing the data to the back end database server.

V. Lesson Learnt/Issues and New Technologies

A. *Lesson Learnt*

New dimension in RIA definitions [5]

a) Instead of relying on caching mechanism such as cookies, HTML5 Web Application (WA) can use client machine native storage, such as local storage, session storage and database storage.

b) HTML5 Web application presents different function through different pages. The browser is refreshed to present the GUI sequences that represent the user workflow and navigation. In RIA, just like in native application the single page application (SPA) is used. RIA runs in one main application window, and user functions are presented through sub application windows within the main window. Only specific sub windows is refresh at a time.

c) Improved presentation layer includes page layout and user interaction and behavior.

d) Support both synchronous and asynchronous communication methods.

Process – challenges in doing large scale JavaScript development

We find that the two most challenging things in working with large scale JavaScript development are modularizing the JavaScript component and performing code quality testing. This is due to the lack of JavaScript tools can be used at enterprise level.

Technical challenges – Browser support / incompatibility

Web developer working with multiple browser vendors across multiple devices is always a challenge. Furthermore, HTML5 specification has just achieved W3C Candidate Recommendation. It would take time for browser vendor to fully compliance with the specification. Therefore, cross browser compatibility issues are not expected to be resolved soon.

Support of workflow engine

Lastly, as the web application can perform at near native application speed, the implementation of workflow engine in web-based RIA is not too far fetching. An example of implementing a workflow in the web application is to improve the interactivity through a managed user behavior (state and transition) approach using JavaScript based workflow.

B. *New Technologies*

Asynchronous Servers – Node.js, Vert.io, Javascript has always been a client-side script since Netscape. With the introduction of Node.JS, Vert.io and other asynchronous server, JavaScript can be extended to server-side scripting. Therefore, this new technologies provide a platform to use a single language for development and enable asynchronous script execution at server-side.

Back End as a Service – or also known as BaaS, used to made available the back-end resources such as cloud storage, user management, push notification and integration with social networking services to the web application. This new platform leverage on the growing needs of web-based RIA requirement to access enterprise resources.

Javascript engine/code generators/optimization – The performance of JavaScript engine has improved drastically since 2010. The modern browser vendors have continued innovating ways to improve performance. Google, as an example, has been improving its Chrome browser's performance with large-scale architectural changes through better use of multi-core processor [12]. On the other aspect, LLVM (Low Level Virtual Machine) JavaScript compiler is available for converting C or C++ code to Javascript.

VI. Conclusion and Future Work

This project show us the potential of combination of HTML5, CSS3 and Javascript to create standardized RIA that can run on different browsers on different devices. Moving

forward we have to further improve our process, tool chain and framework to support scalable RIA development based on these technologies. As the Javascript engine and rendering engine in the browsers further improved we believe these will be the way to create multiplatform application in the future.

Acknowledgment

My acknowledgement goes to all the Staff Engineer in MIMOS for their openness to share their view and knowledge in particular to web technologies. Thank You!

References

[1] Wang, Fei. 2009. The Development of Rich Internet Application Based on Current Technologies. International Conference on Web Information Systems and Mining

[2] https://en.wikipedia.org/wiki/Rich_Internet_application

[3] Urbieta,Matias; Rossi Gustavo; Ginzburg, Jeronimo; 2007. Designing the Interface of Rich Internet Applications. 5th Latin American Web Congress

[4] Lawton, George. 2008. New Ways to Build Rich Internet Applications. IEEE Computer Journal

[5] Preciado J.C., Linaje M., Comai S., and Sanchez-Figueroa F., "Designing Rich Internet Applications with Web Engineering Methodologies", International Symposium on Web Site Evolution, IEEE, 2007, pp. 23-30

[6] Taivalsaari, Antero; Mikkonen, Tommi; et. All. "Web Browser as an Application Platform" , 34th Euromicro Conference Software Engineering and Advanced Applications, IEEE, 2008

[7] Pavlic,Daniel; Pavlic, Mile; Jovanovic, Vladan; "Future of Internet Technologies" , MIPRO 2012 Croatia, 2012

[8] Jacobs,Ian;Jaffe,Jeff; Hegaret,Philip. "How the Open Web Platform Is Transforming Industry". IEEE Internet Computing. 2012

[9] Chris Grier, Samual T. King, Dan S. Wallach. "How I Learned to Stop Worrying and Love Plugins", Workshop on Web 2.0 Security and Privacy, Oakland, CA, 2009

[10] The Lively PartsBin—A Cloud-based Repository for Collaborative Development of Active Web Content

[11] J.C. Preciado, M. Linaje, F. Sanchez, S.Comai - Necessity of methodologies to model Rich Internet Applications

[12] http://www.theregister.co.uk/2013/04/04/google_forks_webkit_as_blink/

[13] Multimedia Meets the Internet: Present and Future

[14] http://www.keeneview.com/2008/06/buzzwords-20-what-is-web-20-what-is-ria.html

[15] Explorative Authoring of Active Web Content in a Mobile Environment

About Author (s):

"Introduction of HTML5 and CSS3 implementation in all the major browsers recently has changed the RIA landscape"

"Web developer working with multiple browser vendors across multiple devices is always a challenge"