From Desktop to Browser Platform: Office Application Suite with Ajax

Mika Salminen
Helsinki University of Technology
mjsalmi2@cc.hut.fi

Abstract

Web applications have usually been less responsive and provided poorer user experience compared to traditional desktop applications. World Wide Web was not originally developed for highly interactive applications and that is why the client-server interaction used in web has not been ideal for highly interactive applications. Ajax (*Asynchronous JavaScript and XML*) is a set of technologies utilizing standard browser technologies to overcome the traditional problems with web applications. It provides means for client-side event handling, incremental user interface modification and asynchronous data transfer between a client and a server. Use of Ajax has allowed new types of applications to be brought from desktop platform to web platform. One class of these new web applications is presented by Google Docs suite which implements the conventional office applications: word processor, spreadsheets and presentations for web browser. Google Docs is used in this paper as an example of new kind of web applications made possible by Ajax. Google Docs is provided as a hosted service which means that it shares some benefits and drawbacks of other applications provided as service. These benefits and drawbacks are discussed in the paper. Also the overall functionality of Google Docs is explained.

KEYWORDS: Ajax, Google Docs, web application, browser platform, SaaS, software as a service

1 Introduction

Traditionally most everyday applications require installation on a specific computer and the executable code they consist of is platform specific machine code and cannot be run on other platforms without recompiling it from source code. The rest of this paper references to these applications with a term desktop application. Now that the WWW has been very largely adopted and almost every desktop computer has a web browser, new class of applications, utilizing the browser as their platform has started to evolve. The biggest differences in this new class of applications are that the user does not have to install the software, it runs directly from the server and the user interface is written in languages that are interpreted by web browser. This means that only the browser is needed to run the program. These applications are referred later by the term web application.

For long web applications were mostly suitable only for applications which did not require much interaction and changes to the user interface view. This was largely caused by the limitations of the browser platform and its programming languages. Web browser was originally developed to retrieve static documents on request and show them on screen, not for complex, interactive applications [4].

Until the recent years many traditional desktop applications could not be replaced by web applications simply because of the browser platform limitations. At least the usability of an application would have been largely degraded if it had been implemented as a web application. Mostly because of the platform limitations there were no widely known or used implementations of certain application types for browser platform. Conventional office suite applications, word processor, spreadsheet and presentation application, are examples of such applications that were too difficult to implement for the browser.

In the recent years new technologies and concepts have been largely adopted to fill the interactivity gap between desktop and web applications. One of these concepts, addressing the lack of interactivity, is called Ajax (*Asynchronous JavaScript and XML*). Ajax is used to improve interactions and data transfer between the web browser and the server. The biggest advantage of Ajax is that it releases web applications from the old client-server interaction mode which required the whole page to be reloaded after every client interaction with the server. This used to make web applications heavy and slow to use and discouraged to implement heavily interactive applications on web platform.

Even before Ajax concept highly interactive applications for web applications could be done by installing plug-ins to web browser. For example Flash applications and Java applets could be used to implement highly interactive applications for web browser. The difference between Ajax and browser plug-ins is simply that plug-ins require installation. The application code written for a specific browser plug-in can not be interpreted on a standard browser out-of-the-box but requires installation of the specific plug-in. Ajax utilizes standard technologies implemented in all the recent browsers. It does not require a plug-in to be installed.

In this paper we will first introduce the Ajax concept and traditional web application interaction model in more detail and discuss the technologies that form the base for Ajax. After that we will explain what kind of benefits Ajax web applications, and web applications in general, have over the desktop applications and also what are the drawbacks. We will focus on web applications provided as a subscription based, hosted service and use Google Docs office suite as an example.

2 Background

2.1 Traditional Web Applications

World Wide Web was originally developed to deliver hypertext documents and that is also reflected to the way traditional web applications work. Almost every interaction with web applications user interface causes an event to be fired which is processed by sending a request to a server. The server processes the request and replies with a document, which the client's browser renders on the screen. [4]

This kind of interaction model might work well for static documents, but it is not ideal for web applications. Every request to the server takes some time and user cannot do anything but wait while the request is processed. [4] For example, after every click of a button the user ends up waiting for the results. Even if only a little part of the user interface needs to be updated as a result of the click, the whole page with the updates needs to be retrieved from server and rendered again on the client's browser.

This interaction mode causes lack of responsiveness experienced in traditional web applications and it is addressed by Ajax technologies which are discussed next.

2.2 Ajax Web Applications

Term Ajax was first introduced by Jesse James Garrett in his article "Ajax: A New Approach to Web Applications" [4]. In the article Garrett defines the Ajax as a set of several technologies for web applications, not as a new technology itself. Ajax incorporates XHTML (Extensible Hypertext Markup Language) and CSS (Cascading Stylesheet) based presentation, dynamic display and interaction using DOM (Document Object Model), XML and XSLT (Extensible Stylesheet Language Transformations) based data interchange and manipulation, asynchronous data retrieval using XMLHttpRequest and JavaScript scripting language to bind all the technologies together [4]. Next, the technologies behind Ajax will be presented shortly.

JavaScript

JavaScript can capture the user interface events from the client which makes the interaction with the web application possible without passing the events to server for processing every time [14]. Events that do not require new data from the server can be processed entirely on the client-side.

Document Object Model

DOM (Document Object Model) is a presentation of hypertext document as an object tree and it allows modifying the document and thereby the presented content by JavaScript on the client's browser. It makes possible to modify the document presentation and data without having to reload the page from server every time a change is made. Modern browsers provide documents as object model for JavaScript. For example elements can be added and removed from page and their properties can be changed using JavaScript and DOM. [13]

Cascading Stylesheets

CSS aims to separate the presentation of document from its contents. It provides the means to describe for example the sizes, positions and colors of elements in a document [11, 2]. These properties are available and modifiable through DOM with JavaScript.

XMLHttpRequest

XMLHttpRequest is an object, accessible by JavaScript, which allows making requests from browser to server on the background, asynchronously without reloading the document after each and every request. Since the request is asynchronous, the browser will not hang while waiting the response for the request. Unlike the name of the object suggests the data can be transferred in any format, including XML, but not as the only format. [9, 1]

3 Office Suite Applications With Ajax

Using Ajax technologies has made it possible to make traditional web applications more interactive and responsive, and thus improved the user experience. E-mail applications with web user interface, webmails, have been available for years and many of them have got a new, more interactive interface as they have been converted to using Ajax. Companies such as Microsoft, Yahoo and Google have introduced Ajax-based webmail applications. Also, web based calendars have existed before and now, for example Google has introduced a web calendar application utilizing Ajax. Thus, Ajax has been used to improve existing web applications.

In addition to improvements for old applications, richer user interfaces achieved with Ajax have been applied to bring traditional desktop-only applications to browser platform. Google has an application suite named *Google Apps* [7] which provides basic office applications as a service used through web browser and their technology is largely based on Ajax. The application suite includes calendar and webmail applications and also a package named *Google Docs* [8] which provides presentation, spreadsheet and word processor applications. This paper uses Google Docs suite as an example when discussing the benefits and drawbacks of Ajax web application approach versus desktop applications. This certain application package was selected because it presents new types of web applications that largely rely on Ajax. The Google's office suite works as a proof-of-concept of what can be achieved with Ajax.

3.1 Google Docs Compared To Traditional Desktop Applications

Since in this paper, we focus on hosted, subscription based web applications such as Google Docs, we cannot omit the fact that many of the features and qualities of the Google Docs are common to all such hosted applications. SaaS (Software as a Service) is the concept where customer buys the service, the possibility to use the software and not the actual software. It distinguishes the ownership and possession of the software from its use [10]. Usually the software is hosted by the SaaS provider and fees are subscription based rather than traditional licensing fees. There is no need to install application client programs. All the software resides on the server.

Google Docs is an example of service based on the SaaS concept. That is why the benefits and drawbacks of SaaS concept are explained first. After presenting the common SaaS case we will discuss in more detail about the Google Docs features, benefits and drawbacks to illustrate the power of Ajax-based web applications.

3.2 Benefits and drawbacks of SaaS concept

Although Google Docs brings new kinds of applications available over Internet, most of the ideas behind it are based on SaaS concept. That is why most of the qualities and features of the software can be examined by looking at what is said about the applications utilizing the SaaS concept. SaaS concept has been researched for years and there exists lots of studies about it [10][6][12]. In this section its quality properties, which apply also to Google Docs are discussed. Andreas Goeldi et al. describe these aspects in their research study and the following listings are largely based on what they have stated in their study [5].

3.2.1 Benefits for the Application Service Provider

Company providing applications for its customers can benefit from utilizing the SaaS approach mostly by reduced deployment, support and maintenance costs.

Since the software is run over the Internet using a web browser, there is no need to install the software to customers computers. As the customer buys the service, SaaS provider needs to possibly only setup some user accounts for the new users. Support of different kinds of client environments and installation support is not required. It is sufficient to support the quite well standardized browser platforms [5]. For example the application can be run on any operating system, hardware or network infrastructure. The only requirements are web browser and Internet connection.

Also the maintenance costs are reduced, because issues that come up with the application can usually be examined and resolved without an expensive visit to a customer. Since all data and software related to the application resides on the server-side, issues can be diagnosed on the SaaS provider's side [5].

Changes such as bug fixes and new features can also be easily deployed to customers. Deploying an update to all customers requires only updating the software on the SaaS provider's servers. Client interaction is not required [5].

3.2.2 Benefits for the Customers

Since starting to use SaaS application does not usually require any new software to be installed on customers' computers, large IT infrastructure is not required either. For self-hosted software customer usually needs to maintain in-house servers and other infrastructure which are not required for SaaS applications. This can bring cost savings and is especially favorable to small companies who do not have existing infrastructure and large IT budget. [5]

Because the SaaS applications do not require any installation to specific machine, it is also very easy for an end-user to move between different workstations and environments. The SaaS based application and data is instantly available as the end-user moves to a different workstation.

The entry costs for new software and deployment time can also be very low in SaaS because of the hosted software and subscription based fees [5]. This can be very good for a small business that does not want to make big investments at the beginning and also reduces the risk of starting to use a new piece of software.

Also the reliability of the software can usually be made better on hosted service than on small company's own servers [5]. The burden needed for supporting high reliability of an application such as frequent backups, redundant hardware, non-interfered and secured electricity supply and constant monitoring is taken away from customer.

3.2.3 Drawbacks of SaaS

Privacy and security can be a huge issue for the customer in SaaS concept. Because all the data is located on the SaaS provider's servers it's privacy and overall security can be trusted as much as the company that hosts the service. For example, many large enterprises require full control over company data and would not likely trust company like Google to have the possibility to access it.

Also, there is the security risk that the SaaS applications could be brought down by a DDoS (Distributed Denial of Service) attack. During the critical business hours this could have devastating financial effects for large group of customers.[5]

The application could also be more vulnerable to external attackers who want access to company data. Because the application is hosted on a server which is accessible from public Internet it might not be as well secured as an application that is deployed to a company's private, in-house, network. For example, easily guessable, weak password could allow attacker to get access to company data.

While the software is hosted on a computer on the network it is available only when the customer has access to the network and also the service is online. Internet access is essential to use the software [5]. It is not possible to get any or at least full benefit from the software if the Internet access is not available. It means that, for example, the software cannot be used in most airplanes and remote locations without Internet access.

The integration of SaaS applications with existing applications is not well supported either. Especially integration with legacy applications behind company firewall is not supported. [5] This is well understandable since the actual application is run on a server located in public Internet and access to private network is usually prevented from there.

One big issue with SaaS applications has also been that they have not been able to provide as good usability, responsiveness and features as traditional web applications. [5]. This is something that Ajax has partially addressed. Next, we will explain the improvements in this area by using the Google Docs as an example.

3.3 Google Docs Overview

What Ajax has brought to web application can very well be seen by looking at Google Docs applications. We have used the applications ourselves and the overall feel of the text processor and spreadsheet applications is quite pleasant. The user interface is responsive, and the screen is not flickering all the time. The applications succeed to bring the look and feel of the normal desktop applications to the browser platform. All the changes to documents are saved automatically on the background so the user does not have to be afraid of network outages. Basic keyboard shortcuts are also supported and it is easy to forget that you are working with a web application.

Still the Google Docs is not perfect office suite. Features provided by the Google's office applications are quite limited. For example the text processor supports only the most basic formatting, there is no formula editor available or other more advanced features. On the other hand this makes the application also easy to use, while there are no tens of menus and toolbar icons where the needed features could be hidden.

Although the Google Docs does not have all the features its desktop counterparts have it has some extra features compared to them. Especially collaborative work in Google Docs is very well supported. Stijn Dekeyser and Richard Watson discuss the functionality of Google Docs from researcher collaboration perspective in their technical paper titled *Extending Google Docs to Collaborate on Research Papers* [3].

Google Docs applications allow users to invite other people to collaborate with a document. Invited collaborators can be given permissions to only view the document or also permissions to edit it. Users can edit the document simultaneously and updates to document are synchronized to other users view approximately every thirty seconds. If multiple users edit exactly the same part of the document, a conflict can occur. If users make a conflicting change to a document, application informs the users about the conflict, shows the conflicting text and reverts the conflicting part back to the state before the conflict. The conflict can further be resolved by selecting the preferred version of content and reapplying it to the document. [3]

Google Docs applications also keep revision history of the documents. User can check how the document looked like at any time past, compare different revisions of the document and revert back to a previous revision. [3] This version control feature, versions the documents automatically on the background so the user does not have to remember to make the revisions.

The Google Docs applications also support exporting into different formats for viewing, publishing and editing. Supported formats include PDF, Microsoft Word document and HTML. This allows user to access their documents even when working offline. [3]

Overall, the Google Docs is a simple application suite that has the most essential functionality of office suite applications. Although it is probably suitable for most needs, it cannot be seen as a total replacement for traditional desktop office suites, because it is missing some advanced functionality.

4 Conclusion and Future Work

We discussed why traditional web applications cannot deliver the same user experience than desktop applications. This is largely caused by the fact that the interaction model with the server is not ideal for web applications and requires retrieving and rendering the whole view on user's browser even though only a small part of the screen would really need to be updated as a result of a user interaction. Transferring data from server to client can only be done by passing a new HTML document presenting the data to the client.

With Ajax technologies web application's interaction with server can be made more efficient. Standard browser technologies: XHTML, Cascading Stylesheets and JavaScript programming language are utilized to update the user interface view without requiring to retrieve and render the whole view after every user interaction. Changes to the user interface can be made incrementally by modifying the Document Object Model tree and data can be passed, on background, in any format between the server and client using XMLHttpRequest object. The user event handling is made using JavaScript.

Ajax has allowed improving the interactivity and responsiveness of old web applications but it has also allowed implementing some new web applications of which Google Docs is an example. Google Docs is an office application suite which includes basic word processor, spreadsheet and presentation applications. The suite cannot compete with desktop applications in terms of functionality, but it includes some new features especially good for group collaboration. User experience of Google Docs is already very close to desktop applications, thanks to Ajax.

We also noted that since Google Docs is provided as a hosted service rather than an installable application it shares many benefits and drawbacks with other software utilizing the SaaS concept. Most of these benefits come from taking the burden of maintenance away from customer, allowing easy access and entry for the software, easier support and software update deployment for the software provider. Drawbacks of this approach are caused by the fact that all the data and software resides on the software provider's servers. Servers are vulnerable to DDoS attacks and software user needs to trust the provider to keep his or her data private. Also the need for network connection can be seen as a drawback because it is not available all the time everywhere.

Based on the Google Docs it seems that it is possible to achieve nearly desktop application like user experience with a browser platform application. There are also many benefits that this platform provides, but they do not come without drawbacks.

In the future we would be interested in a study that would elaborate the real contribution of Ajax to the web. Since Ajax technologies existed before the actual term was introduced, it is not very clear what is the new thing it has really brought. Has it advanced the standardization of technologies it is based on? Is Ajax just a product of a political manifesto, which "legalized" JavaScript, that was previously considered bad among web developers?

References

[1] Anne van Kesteren. The xmlhttprequest object, April 15, 2008. Accessed: April 15, 2008. http://www.w3.org/TR/XMLHttpRequest/.

[2] B. Bos, H. W. Lie, C. Lilley, and I. Jacobs. Cascading style sheets, level 2 css2 specification, May 12, 1998. Accessed: April 14, 2008. http://www.w3.org/TR/1998/REC-CSS2-19980512/.

[3] S. Dekeyser and R. Watson. Extending google docs to collaborate on research papers. Technical report, The University of Southern Queensland, Australia, 2006. Accessed: March 16, 2008. http://www.sci.usq.edu.au/staff/dekeyser/googledocs.pdf.

[4] J. J. Garrett. Ajax: A new approach to web applications, 2005. Accessed: February 8, 2008. http://www.adaptivepath.com/ideas/essays/archives/000385.php.

[5] A. Goeldi, T. B. Jones, and B. Lo. Google apps for enterprise installed solution. Technical report, MIT Sloan School of Management, December 2006. Accessed: March 14, 2008. http://tbjinvestments.typepad.com/tbj_investments_llc/files/15_567_google_enterprise_installed_solution_goeldijoneslo.pdf.

[6] N. Gold, A. Mohan, C. Knight, and M. Munro. Understanding service-oriented software. *Software*, 21(2):71–77, March-April 2004. Accessed: April 15, 2008. http://ieeexplore.ieee.org/iel5/52/28453/01270766.pdf?tp=&isnumber=&arnumber=1270766.

[7] Google Inc. Google apps. Accessed: February 11, 2008. http://www.google.com/a/help/intl/en/business/applications.html.

[8] Google Inc. Google docs. Accessed: February 11, 2008. http://www.google.com/a/help/intl/en/users/user_features.html.

[9] D. McLellan. Very dynamic web interfaces, 2 2005. Accessed: April 30, 2008. http://www.xml.com/pub/a/2005/02/09/xml-http-request.html.

[10] M. Turner, D. Budgen, and P. Brereton. Turning software into a service. *Computer*, 36(10):38–44, 2003. Accessed: March 14, 2008. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1236470.

[11] W3Schools. Introduction to css. Accessed: April 30, 2008. http://www.w3schools.com/Css/css_intro.asp.

[12] B. Waters. Software as a service: A look at the customer benefits. *Journal of Digital Asset Management*, 1(1):32–39, January 2005. Accessed: April 15, 2008. http://www.ingentaconnect.com/content/pal/dam/2005/00000001/00000001/art00007.

[13] World Wide Web Consortium. Document object model (dom). Accessed: April 15, 2008. http://www.w3.org/DOM/.

[14] World Wide Web Consortium. Javascript events. Accessed: April 14, 2008. http://www.w3schools.com/js/js_events.asp.