Does it make sense using Ajax for a Dynamic XML Table-Editor?

Sebastian Skritek
Wolfgang Aigner
Silvia Miksch

Vienna University of Technology
Institute of Software Technology & Interactive Systems (ISIS)

Asgaard-TR-2005-12 December 2005

Authors: **Sebastian Skritek**
Wolfgang Aigner
Silvia Miksch

sebastian.skritekgmx.at, {aigner, silvia}@ifs.tuwien.ac.at
http://ieg.ifs.tuwien.ac.at

Contact: **Vienna University of Technology**
Institute of Software Technology & Interactive Systems (ISIS)

Favoritenstraße 9-11/188
A-1040 Vienna
Austria, Europe

Telephone:	+43 1 58801 18833
Telefax:	+43 1 58801 18899
Web	http://ieg.ifs.tuwien.ac.at

Does it make sense using Ajax for a Dynamic XML Table-Editor?

Sebastian Skritek, 0226286
Sebastian.Skritek@gmx.at

November 2005

3.0SE Seminar(mit Bakkalaureatsarbeit)
a. o. Univ. Prof. Dr. Silvia Miksch
Dipl.Ing. Wolfgang Aigner

Abstract

Ajax is an approach to create user interfaces for web applications that behave like desktop applications. We discuss, whether Ajax is applicable on the Dynamic Query Table Editor, a web application that shall be developed within a practical course. We summarize pros and contras of Ajax and what should be considered when using it. Based on extensive comparison of possible features for the user interface with and without Ajax we conclude that Ajax is a good choice to enrich the application.

Contents

1 Introduction

Ajax (shorthand for Asynchronous JavaScript and XML) is a proposal for how to combine existing technologies to build web applications that react to inputs like desktop applications [Garret,2005a]. Ajax combines the use of the XMLHttpRequest for asynchronous communication with the server, and DHTML to make changes on the site actual shown in the browser. With these technologies, it is not longer necessary to reload complete pages, just because some data should be sent to or received from the server. This allows a user to work without the interruption of a page reload and avoids blank page while the data reloading process. Ajax is therefore regarded as one of the so called "Web 2.0" technologies [O'Reilly,2005; LaMonica,2005; Bryant,2005], and became very popular within the last year. One of the basic ideas of "Web 2.0" is to regard the web as a platform for application developement. Ajax supports this idea by enabling the developement of (operating-) system independet applications, that execute within the browser. Two famous and widley referred web applications using Ajax are GMail [Google,2005a] and Google Maps [Google,2005b], but there are several more, and their number is still growing. A more detailed description of Ajax is presented in section 2.
Neither the technologies combined in Ajax, nor the idea of using them together are something new. In section 3 we gave a short overview about the discussion whether the creation of the name, "Ajax", makes sense, what is critiziecd by a lot of developers, and why Ajax became so popular only recently.
The main purpose of this paper is to discuss whether Ajax can be used for implementation of the Dynamic Query Table Editor (DQT). DQT is intended as a web application for managing information collected on certain techniques for information vizualization. An introduction to the specifications of DQT is given in section 5. It shall be implemented during a practical course.
In section 4 we summarize the advantages and disadvantages of Ajax. Additionally we suggest what should be considered or avoided when using Ajax, and evaluate these suggestions for DQT in section 5. In general, the use of Ajax should be critically considered and planned, and plans should exist for what happens to people whose browsers do not support Ajax. It should also be considered whether the use of Ajax does improve the application for the user or not. For this, in section 5, we compare two solutions for the user interface of DQT, one implemented using the Ajax technologies, and the other implemented traditionally, that is without the Ajax-technologies.

2 Ajax

The term "Ajax" was introduced by J.J. Garrett [Garret,2005a], and is the acronym for "Asynchronous JavaScript and XML". Ajax is not a new technology, but a distinct set of current technologies and about how to use them together for creating web applications. Therefore, Ajax describes a design approach, and could be called a design pattern, too. The used technologies and their suggested functions are:

- XHTML and CSS for a standardized layout
- the DOM for a dynamic interface and responses to user input
- XML (and XSLT) as data transport (and manipulation) format

- XMLHttpRequest for asynchronous communication
- JavaScript for combining all together

The idea of Ajax is to overcome the behavior of classical web applications. In these, most of the user actions (like submitting data inserted into a form or requesting more data) cause a request sent to the server. The server processes the data and returns a new HTML page to the client, whose browser replaces the old page with the new one. Because of this behaviour, the user application is blocked while it waits for the response, and the workflow is interrupted.
Using Ajax, it is possible to build web applications that behave more like desktop applications. That is, they are able to react immediatly to user inputs and provide a continuous user interface during the whole session without reloading. Therefore DHTML allows to react on user input immediatley and without reloading the whole page, so the user may not have to stop working. If it's necessary to communicate with a server, this can be done asynchronously (with respect to the interactions of the user) using the XMLHttpRequest. So the user again can continue working (as long as the server response is not essential for continuing work). Again in combination with DHTML, the page does not need to be completly rebuild when the response arrives.
The differences between the models of traditional and Ajax-Based web applications are shown in figure 1.

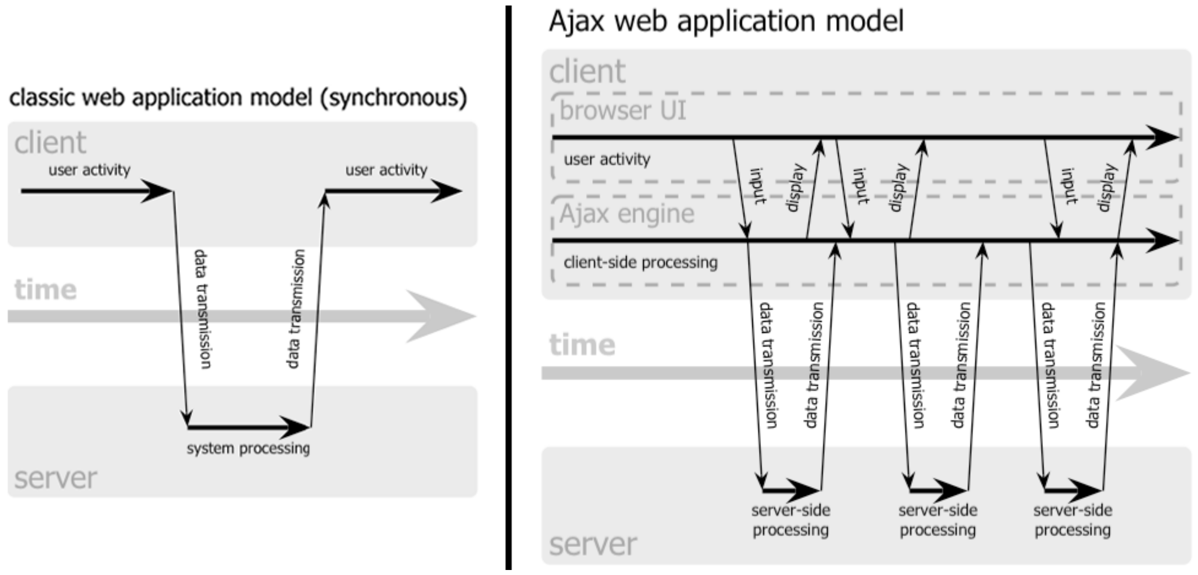

Figure 1: Differences in the (communication) models of classic and Ajax web applications. Note the additional layer (the Ajax engine) at the client. [Garret,2005a].

Although in the first publication [Garret,2005a] assumed primarily XML data to be sent via the XMLHttpRequest, in fact data of arbitrary format can be transferred.

2.1 Why only now? - Reasons for recent use of Ajax

Neither the technologies used by Ajax, nor the idea of putting them together is completely new. Even the latest of the used technologies, XML and XSLT, are about 6 years old, and the most essential parts, DHTML the XMLHttpRequest are even available for

some more years. They also have been used together [Massy,2005]. So, except for the name, Ajax isn't something completely new, a fact that many people like to point out [Massy,2005; Obasanjo,2005a; Papageorge,2005b]. Giving it a name is also sometimes critizied [Hixie,2005], and there are also demands for using a more meaningful name (like "XMLHttpRequest"). But there are also good reasons for naming it explicity, as well as there are reasons why it became so popular only recently:
Having a catchy name for it helps to communicate the idea of Ajax because of two main reasons: First, it makes it easier for the community (i.e. developers, programmers, etc.) to simply talk about. Instead of complicated explanations of what we actually want to talk about,

> *"[..]we can say "Ajax" and move on to more interesting things."* [Willison,2005]

The other reason is, that a catchy name helps making Ajax known, not only within in the community, but especially for people without a technical background. Ajax simply sells much more better than for example "remote scripting" or "XMLHttpRequest" [Haughey,2005].
Another reason that helps making Ajax known is when there are applications available in the web, that are quite popular and using Ajax, because then it's possibly to simply say "this is Ajax". Such applications are available now, like GMail [Google,2005a], Google Maps [Google,2005b], or Flickr [Flickr,2005], that use Ajax at least to some extend. One more reason why Ajax becomes so popular right now, is, that support for the technologies incorporated is now available in nearly all browsers, most of the time even according to the existing standards or W3C recommandations. There are also code libraries available now, as well as design guidlines and patterns, that makes it easier to create Ajax application.
Also, there's now an ongoing trend towards using browsers (and the web in general) as a platform for applications [Rees,2002; O'Reilly,2005; Lopp,2005; LaMonica,2005]. The advantage of browsers is, that they are available on nearly all systems. This provides the possibility to develop applications independent of the underlying system. This idead isn't new either, as it was one of the intensions of Netscape for releasing JavaScript [Champeon,2001]. But this idea to use the web as a platform achieves only now, sometimes refered to as "'Web 2.0"' [O'Reilly,2005]. As Ajax enables building web application that nearly behave like desktop applications, and because of the possibilities it gives to developers, Ajax became quite popular within this background [LaMonica,2005].

3 Underlying Technolgies

In this section, we shortly review some of the underlying of Ajax, and what's their suggested use.

3.1 JavaScript/ECMA Script

JavaScript (JS) was developed by Netscape together with Sun Microsystems, and was introduced in 1995 with version 2.0 of the Netscape browser [Champeon,2001; Ginda,2004; Wilkens,2002]. JS is an interpreted, object-oriented programming language based on prototyping. It was intented to enable easy control for Java Applets, and played an important role in Netscape's idea to introduce the browser as a platform independent from the underlying operating system, to enable consistent web browsing and developing

[Champeon,2001]. However only manipulation of the content of the HTML pages became the main use for JS. Because of incompatibilities between JavaScript and JScript (Microsofts counterpart to JS, released with IE 3.0 in 1996, [Microsoft,2005a]), there was an attemp for standardaziation. A standard was released by the "European Computer Manufacturers Association" (ECMA)[ECMA,1999], referred to as ECMAScript. Different languages supported by different applications (i.e. JavaScript in browsers that use the Gecko engine [Baron,2004], JScript in the IE, ActionsScript in Macromedia Flash and more) claim to be compatible with this ECMA specification. When referring to web browsers, JavaScript is often used as acronym for ECMAScript. The latest version is ECMA-262 Edition 3 from 1999, which is implemented in JavaScript 1.5 [Ginda,2000].
In Ajax, ECMAScript is one of the core technologies: It does not only provide a programming language for processing of received data and user inputs, but also DOM and XMLHttpRequest are accessible only through ECMAScript.

3.2 DOM

The Document Object Model (DOM), is an interface to access and manipulate documents, like XML or HTML documents [Hegaret et al.,2005; Hors et al.,2004]. In DOM, the tags of a document are represented as nodes arranged in a tree. For navigation within or manipulation of the tree, the DOM defines specific operations. The DOM was introduced with the versions 4.0 of Netscape's an Microsoft's web browsers [Champeon,2001]. Because of many incompabilities between these two proprietary DOMs, the W3C standardized the DOM, so that it now offers a uniform API for working with XML.
Although HTML pages, in general, are no well formed XML documents, most of the browsers offer the possibility to make dynamic changes on the current page, by offering access to a DOM representation of the page. In general, this can be accessed, hence manipulated, using the browsers implementation of ECMAScript, what's refered to as DHTML [Schaefer,2005]. In Ajax, the DOM is used in combination with DHTML and to process incoming XML messages.

3.3 XMLHttpRequest

The XMLHttpRequest offers within a browser the possibility to communicate asynchronously (with respect to user interaction) with a server [Microsoft,2005b]. It provides an API, that can be used (by the browsers implementation of ECMAScript) to make a HTTP-request without reloading the page. The API enables both, specifying some properties of the request as well as access to the data received by the server response. When the response arrives, a function, specified before sending the request, is called by the request object. Despite its name, data sent using the XMLHttpRequest needs not to be XML formatted. The actual format can be arbitrary.
The XMLHttpRequest was invented by Microsoft with IE 5.0 as an ActiveX Object [Apple,2005], but in the meantime, most of the browsers provide the same funtionalitiy accessible through the same API [Zammetti,2005].
Within Ajax, the XMLHttpRequest provides the ability for sending or retrieving data asynchronously.

3.4 XML & XHTML

Extensible Markup Language (XML) is a W3C recommendation for a markup language [Quin et al.,2005; Thompson et al.,2005]. The current version 1.1 was released in 2004 [Yergeau et al.,2004s], but users are encouraged to use to use version 1.0 [Yergeau et al.,2004a] if the new features of XML 1.1 are not needed. It is a subset of SGML [Mason et al.,2001], and can be used to describe and structure all kinds of data. It's often used as a format for exchanging data between different systems, because it's readable for both people and machines, and can be processed efficiently. That are also reasons why it's the suggested format for Ajax applications. Another reason is, that there are quite a lot of tools for processing XML data. For example, in the browser the DOM can be used to handle received data. XHTML is a specification to make HTML documents valid XML documents [W3C,2005].

4 Aspects of working with Ajax

Before we evaluate Ajax for implementing the DQT, we give an overview of advantages and disadvantages of using Ajax in general, and present some guidelines that shall help to build useful applications with Ajax. None of these lists claim to be complete, as this, if possible at all, would exceed the scope of this work.

4.1 Advantages of using Ajax

As mentioned in section 2, Ajax enables web applications to behave more like desktop applications. Such web applications are also referred to as ”rich internet applications”. Besides Ajax other technologies for creating such web applications are for example Flash and Java.
Therefore, Ajax supports the creation of much more responsive, interactive and stable (that is no reload) user interfaces. This is not only restricted to the possibility for creating animations, but also for allowing to use drag and drop, or to replace or change parts of the page dynamically, what can be used for giving immediately very adequate response to the user. [Mahemoff,2005; Garret,2005a]
With ECMA Script, there's a programming language available on the client machine, that allows processing of user input directly at the client.
By the possiblities of changing the DOM and making requests to the server in the background, reloading the whole page is no longer necessary, what besides the improvement of usability leads to a decrease of network traffic while the application is running. [Fried,2005; Mahemoff,2005].
Ajax enables keeping the actual visited page up-to date, by polling the server and updating the information on the page (”Real time update”) [Mahemoff,2005].
The big advantage of Ajax over technologies like Flash or Java is, that it does not require any browser plugins, as all of the major browsers provide native support for the applied technologies. Additionally, Ajax allows to develop browser independet applications, that behave identically on the majority of internet-browsers [Bergmann and Bormann,2005].

4.2 Disadvantages of using Ajax

Using Ajax, however, has some disadvantages that should be considered as well [Grossman,2005].
Ajax do not work with all browsers, for some reasons: First, certain features needed for Ajax (like JavaScript or the XMLHttpRequest) might be disabled (for security or other reasons). Another reason can be, that the browser used is too old, or simply does not support this technology.
Additionally, although Ajax is based on standards, theses standards are not implemented completly and without errors in all browsers [Bosworth,2005a].
Loading the application might take longer as the ECMAScript code needs to be transported to the client additionally to the content of the website.
The browser's "back button" and "bookmark functions" do not behave as expected, as calls to ECMAScript do not produce an entry in the browser-history [Stenhouse,2005; Neuberg,2005].
Search engines can't index the parts of a site that are loaded dynamically.
Because the application logic is distributed on server and client, developing the application becomes more complicated, hence more error-prone and expensive.

4.3 Problems to be avoided and aspects to be considered when using Ajax

Quite a lot of suggestions have been made what should be avoided or considered when creating Ajax applications. We can only present an assortment of them, focusing on aspects that are important with respect to our application, or important general rules.

- The use of Ajax should be thoroughly planned. Ajax should not be used for the sake of itself, but the user should gain from the use of Ajax. It might be useful to be clear about whether the product is more like a website, or a web application. Once decided to use Ajax, it should not be overused within the application or website [Garret,2005a; Rees,2002].

- There should be alternatives for users whose browser does not support Ajax or parts of it. Ideally, a version of the site or application should be available that works without ECMAScript, hence without Ajax. For most of the applications, only telling the user that JavaScript is needed to visit the site, is not satisfactory for the user [Baekdal,2005; Garret,2005a] (although a popular practice, and even used by [Google,2005b]).

- The user should not need to wait for Ajax. One of Ajax advantages is that it overcomes the page-reload. The use of Ajax should not bring in the aspect of waiting for a server response again, if it's not inevitable. For example, checking data inserted to a form against the server or loading additional information according to the input, performed in background, while the user finishes filling out the form, can be a benefit for the user. But blocking the "submit-button" until all requested data have arrived probably leads to user frustration [Bosworth,2005a].

- The use of Ajax should not interrup the workflow of the user. That is, the user should not lose the focus on what he's actual doing, because some parts of the page changed (unexpected, i.e. as a result of an asynchronous request). In general, changes of the page should not happen unexpectedly, or in a way that the user needs to re-orientate or search for where s/he was before the update took place [Baekdal,2005; Bosworth,2005a].

- The user should keep some kind of control about what happening with the data s/he inserts. For example, if a user inserts data into a form, this data should not be saved automatically, but only if the user requests that [Bosworth,2005a]. This is because of two reasons: First, the user is used to editing a form locally does not automatically changes the data stored in the database, which s/he might wants to remain unchanged. Secondly, the user knows when s/he has finished inserting, hence when the data is ready for sending. Doing this automatically would lead to a lot of data sent to the server for saving, just to be overritten after the next change the user makes to the form.

- The user interface should satisfy existing UI conventions, either to ones for applications or for websites, depending on what can be done on the page. The idea is that it's easy for people to use the site, without learning to handle the interface takes too long, and without any difficulties. The user should always have a clue of what happens in response to her/his inputs [Mahemoff,2005; Baekdal,2005].

- Because the browser does not give any response to the user when s/he triggers an Ajax event, the site or application itself must give immediate visual response to user inputs [Bosworth,2005a].

- When it makes sense, techniques should be used to keep the functionality of the browser's "back button" and "bookmark functionality". That is, when taking one step back, or accessing a site directly, leads to a valid state [Neuberg,2005; Bosworth,2005a].

- Although Ajax is based on standars, that should behave identically on all kinds of clients, Ajax applications need to be tested on the different clients, espessially on a variety of browsers, to ensure identical behavior on all all of them. Therefore, although conceptually Ajax is a single platform development, using Ajax should be treated as multi-platform development [Baekdal,2005].

- The use of libraries or frameworks is strongly recommended [Bergmann and Bormann,2005].

- Data sent using the XMLHttpRequest are, without further actions taken, sent as plaintext. So privacy should be an important issuse of the development of Ajax applications [Baekdal,2005].

- Data traffic should be minimized [Mahemoff,2005; Bosworth,2005a]. Data should only be loaded if needed (except for precaching to provie faster response to user inputs). When using some kind of polling for keeping information up to date, it should be considered what this implies on the used bandwidth and server loads, especially when the site or application is visited by many people simultaneously [Almaer,2005]. It should be further considered whether Ajax is the apropriate method for loading entire pages.

Concrete technical design patterns can be found in [Mahemoff,2005].

5 Ajax features for the Dynamic Query XML Table-Editor

5.1 The Dynamic Query XML Table-Editor

The Dynamic Query XML Table-Editor (DQT) shall become a web/database application, that helps managing and accessing collected data. The specific reason for developing the DQT were data about the features of techniques, that were stored in tables like table 1, formatted and saved as LaTeX files. Each entry in these tables is classified by a set of

	attribute 1		**attribute 2**		**attribute 3**		**attribute 4**		**attribute 5**		**attribute 6**		**attribute 7**			**attribute 8**	
	value	value	value	value	value	value	value	value	value	value	value	value	value	value	value	value	value
entry(Method 1)	●			●	●		●	●			●			●	●		
entry(Method 2)	●			●	●		●	●			●			●			
entry(Method 3)	●	●	●		●		●				●			●	●		
entry(Method 4)	●		●		●			●			●		●	●	●		

● ...completely fulfilled, ○ ...partly or implicitly fulfilled

Table 1: Example of the existing LaTeX-tables

attributes, taht are combined to groups.
Main disadvantage of this storage format was, that it is quite unhandy. To make changes to the data, or select and merge special attributes or methods from different tables is very prone to errors. Sharing those data with others is complicated, and there is no support for searching specific methods (that fullfill certain criteria) either.
DQT shall become an application created during a 10h practical course, that stores the data in a database, and provides an interface for easy inserting, editing and deleting ("inserting/editing functionality"). Besides, it should be possible to look for methods/techniques that meet given requirements (like an attribute having a special value), also supported by an intuitive interface ("search functionality"). Furthermore, there should be a way to export the results to LaTeX again, although the export mechanism should be flexible enough to make adding export-functionality to other formats easy. Therefore the result of the queries is XML coded, and is tranformed to the output format using XSLT. Another demand on DQT is, that it's accessible through the web, and hence provides a simple user management system for restricting access to some of its functionality (like changing the data). The design of DQT should make the adding/removing of attributes easy, as well as to adopt the system for similar problems.

5.2 Evaluation of the application attributes with respect to Ajax

In order to decide finally, whether to implement DQT using the Ajax approach or by the classical one, we tried to work out the differences between the resulting applications. We also considered the problems and guidelines mentioned in Section 4. The results of this evaluation are presented first:

- Because time for developing and implementing, as well as time for testing increases when using Ajax, the required time for te course will increase too. Implementation time increases because of parting the buisness logic between client (ECMAScript) and server (PHP). Because of this it's necessary to send the server-output to the client and process it there, what is one step more than only passing the formatted server-output to the browser. Testing expense for the application when implemented with Ajax will also increase, because of the need to check the client code on different browsers, what is quite more work than just verifying the server-output is displayed correctly in all browsers. Because of this, the possibility that hidden errors remain in the application increase. To reduce testing efforts, there was an agreement to focus developement on only a handful of browsers (IE, Firefox, Safari, maybe one more).

- After a short inquest we arrived at the conclusion that we won't use any library, because it would take quite an effort to learn the correct use of them (as documentation is mostly poor), without giving any advantage.

- Developing a version for users whose browser does not support Ajax-technologies is not necessary, as the content stored and presented by the application is directed only to professionals of informatics, of which we assume that they have both, up-to-date browsers as well as knowledge of how to enable the browser version of ECMAScript.

- For the inserting/editing-functionality, asynchronous communication would not give a significant speedup, because of the small amount of data that could really be transmitted asynchronly. After finishing insertion of information to the form and requesting to save them, the user had to wait until all the data has been sent at once, with or without Ajax. For the search-functionality of the application, the XMLHttpRequest could be used to display the current result of the query given so far.

- Optimizing the search-functionality (e.g. by using some kind of caching) would exceed the scope of the course. The functionality must be implemented naive (that is, each request is passed to the database).

- Using the browser's back button within the parts of the application (editing and searching) would no longer make sense. Within the editing part, going back (to the state before saving) would lead to an illegal state, that is not corresponding with the database, and within the search tool, there's nothing one could go back to. For enabling the back button between the parts of the application, they are located under different URLs, and changing between them will lead to a page reload. As every link in the menu of the page loads a completly new page, and links or actions within a page triggers ECMAScript (hence Ajax) events, handling the application stays consistent for the user.

- The privacy issues for the data transport are not considered, that is the data is sent plain text, as otherwise the scope of the course would be exceeded dramatically.
- As Ajax is mainly on improving the handling of web-application, and not on improving some technical characteristics, the main difference in the application developed with or without Ajax lies in the user interface and its properties. Therefore, in the next subsection, we present a comparison of two suggestions for a user interface for the application. One is for a traditional user interface, the other is for an interface created using Ajax.

5.3 Comparison of UI and application handling

As stated before, the main purpose of Ajax is to create easy to use and powerful interfaces for web-applications. Therefore, we present the results on a comparison between our suggestions for the interface with and without the use of Ajax in more detail. The suggestions of the layout without Ajax were created for the practical course as first high-level design-suggestions before the use of Ajax was considered. So they were created really independent of the idea of Ajax. In fact, we tried to create a design that uses client side scripting only if really necessary. The design of using Ajax was developed during the work for this presentation, based on the results presented above. The main focus of these suggestions lies on how could Ajax be used within the project to create a more usable application, and what would be the differences to the traditional design. We split the application into two parts, according to the two main features: One containing the functionality for managing the data, and one for the search, read and export functions.

5.3.1 Differences on data management

Without Ajax:
Using a traditional design, managing the data would consist of two main pages: One where all entries stored in the database are displayed and can be deleted or chosen for editing, as shown in figure 2a. Another where the attributes of a chosen (or new) technique/method can be set and saved, similiar to the one shown in figure 2b. In figure 2a, when selecting an entry for editing or inserting a new one, the user comes to the page in figure 2b. After finishing inserting the data, when the user wishes them to save, the information is sent to the server where it's checked. If the input is illegal, the server returns an editing site back to the user, on which the inserted data is filled in, and the erroneous fields are marked. When all input is according to given rules, the data is written into the database, and the user receives a site telling her/him that everything worked fine.
There are some problems using such an interface: The user can only add entry after entry, and has to wait until inserting of one entry is completley finished before s/he can continue adding the next entry. Especially the period s/he waits for the data going to the server, gets processed, and returns as a response to the client is simply lost time for the user.
Another problem is, that the page is not clearly laid out, and that it's very difficult (or even impossible) to get an overview over all the data, especially when there's a large number of attributes. Recognizing an error made while inserting the data is not very likely. And, while editing one entry, this entry is isolated from the others, while beeing able to have a glimps onto the other entries might be a benefit.

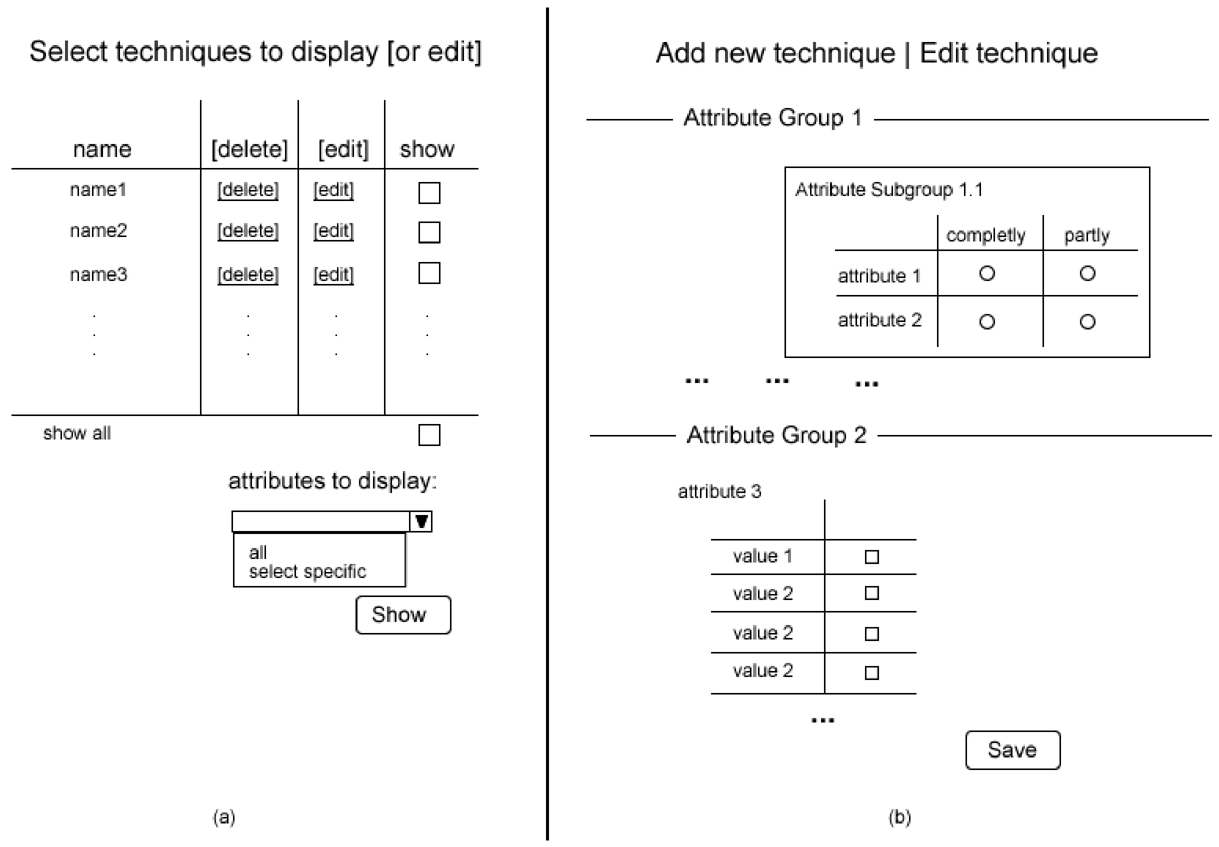

Figure 2: Proposed example for the selection (a) and inserting page (b) without Ajax

Using Ajax:
With Ajax, for managing the data there could be an interface similar to the draft shown in figure 3. The suggested layout for managing the stored data is very similar to the layout of table 1. The values can be changed depending on the possible values of the attributes. Values of attributes that can only have one out of a small number of possibilities (like

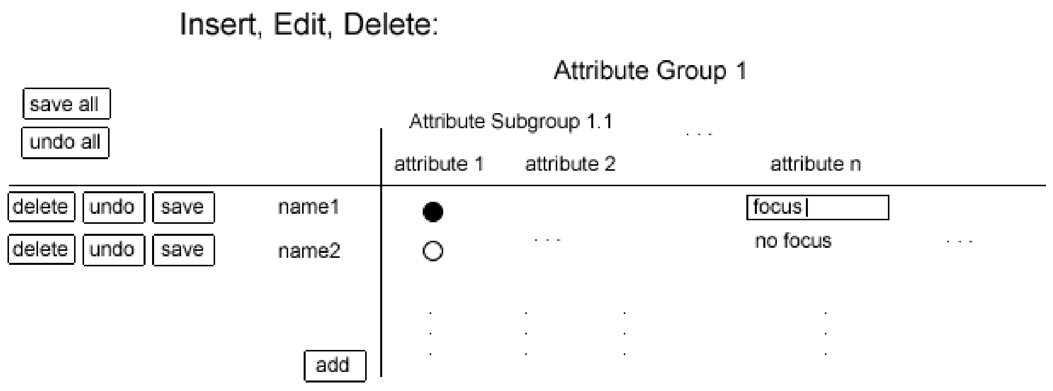

Figure 3: Proposed example for the inserting/editing-page using Ajax

"fully fullfilled", "partly fullfilled","not fullfilled") can be changed by just clicking on the table-cell. By this, the next value of the set is applied (like attribute1). For attributes with values that can be choosen freely (like string or integer types), an adequate form element ist provided within the table (attribute n). For inserting a new technique, a completly new row can be added to the table, by choosing "add".
The layout allows to change the data and to take a look at them at the same time. Because of this, the user sees the result of its input immediatly in table format and within the context of the other entries. This allows the user to keep overview. It also provides a consistent view onto the data, that is independent on whether they are viewed or changed. By using asynchronous communication with the server, data can be saved without the need to interrupt the work. So the user can easily save a data set that s/he has finished, or decide to save the current status, and go on working immidiatly afterwards, without losing the focus of her/his work. To reduce network traffic and to give the user control over the data actually saved to the database, saving is only done when the user explicitly requests it.

5.3.2 Differences on View, Search and Export

Without Ajax:
To view the stored entries, the user starts at the list of available entries shown in figure 2a. Using the checkboxes, s/he selects for which of them s/he wants the application to show detailed information. S/He can also choose whether s/he wants to have all attributes displayed, or whether s/he wants to select certain attributes that should be displayed. Depending on this choice, the server sends a page for selecting the attributes to display (figure 4a), or directly the page containing the selected entries (figure 4b). To select the attributes that shall be displayed, the user has a list of the attributes, structured according to the groups of properties (figure 4a). An attribute is selected by its checkbox, whereupon ECMAScript could be used to update the selection of all sub-attributes according to the

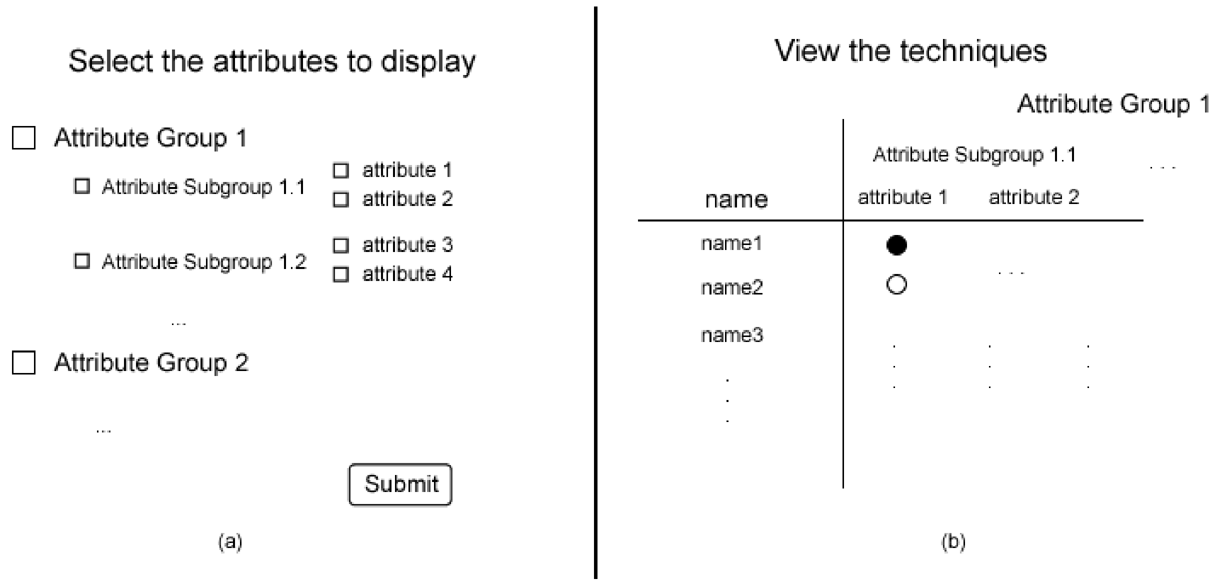

Figure 4: Proposed examples for the page for attribute selection (a) and the table used for presenting the stored techniques (b).

chosen value of the high level attribute.
The presentation of the entries is similar to the representation in table 1, as shown in figure 4b.
For the implementation of the search-functionality, DHTML was chosen anyway to create the inteface for defining the search query. Figure 5 shows a draft for the search form.

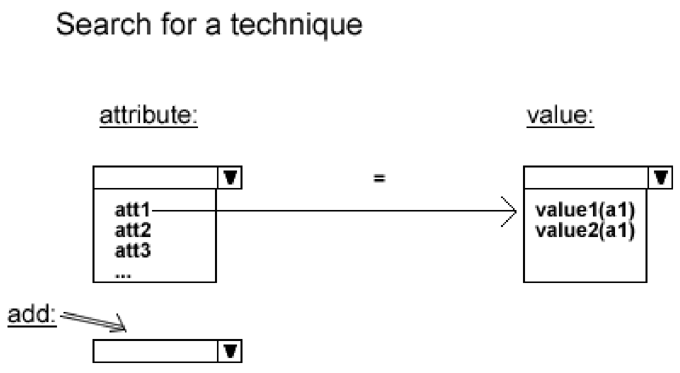

Figure 5: Proposed example for the search form (the main part of the search functionality) without Ajax

A query could be defined choosing an attribute in the left column. Depending on this selection, a form element for specifying the desired value of the attribute is displayed, depending on the possible values of the attribute (that is a textfield for an attribute having a string value or a list of possible values). To add a new row (representing a new constraint), the user chooses "add", that inserts a new row to the page, whereby the user can decide whether this constraint should be linked by AND or OR to the existing constraints. The user can also select the attributes s/he wishes to get displayed on the result, using the same form like the one shown in figure 4a. After the user specified all requirements on the technique wanted, s/he sends the form data to the server. There a database query is created out of this data, and the result of this query are shown on the same page like the one in figure 4b. Once more the problem is, if there are too many (or none) techniques in the result set, the user has to go back and redefine her/his query. Although the form necessary could be inserted to the bottom of the page (so that the results of the old query can be seen while making the changes), to receive the result of the actual query completly interrups the action of defining the query. Especially for using the search functionality for browsing the database (e.g. for getting an overview over entries available, or when the requirements for a technique are not very concret), when the query is probably often redefined, this kind of search functionality is very inefficient and probably quite frustrating.
For the export of results into another format, below the table presenting the actual selection, there's a form where the export-stylesheet can be specified. Afterwards, a page with the result of the export is shown.

Using Ajax:
When using Ajax, we suggest to use only one page for searching and displaying the results as well as for managing the export of the selected data. This page could look like as shown

in figure 6. For specifying the search constraints, we suggest the use of a table similar to

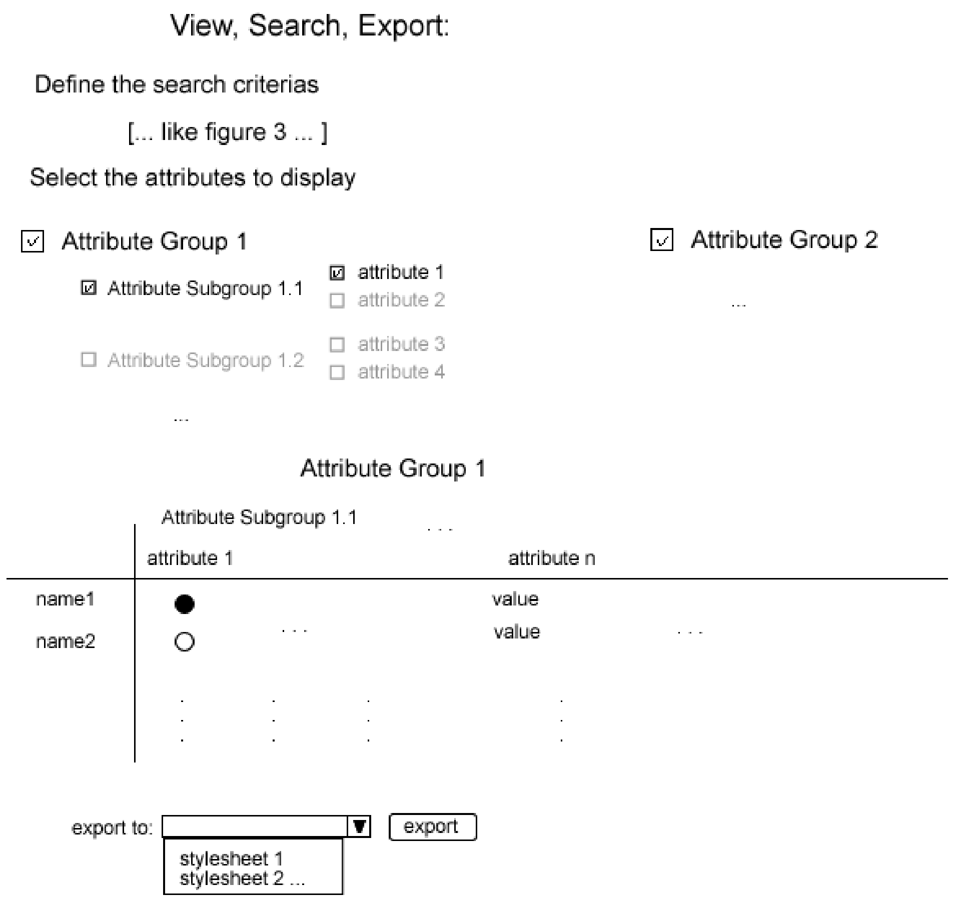

Figure 6: Proposed example for the interface of the search and export functionality using Ajax

the one used for managing the data (figure 3). Within this table, the desired properties can be set. Then all techniques that fullfill the requirements are returned by the server. Therefore, all constraints within a row are put together using AND, whereas the different rows are combined using OR.
This again allows to keep the representation of data for the user consistent, which makes it more intuitive to handle. This should make it easier to put the needs into a query. On the other hand, it should be easier to follow the meaning of the query inserted. Additionally, this should also provide an easier way to refine the actual query according to the results.
The selection of displayed attributes is done the same way as in the design without Ajax, but without the need to load and send an extra page, as the form is now included into the page.
The result of the most recent query is shown in a table on the same page, which has the same layout as the table in the non-Ajax version (figure 6 and figure 3). But with Ajax, one can select certain entries in the table that shall be hidden. Also, both, changes on the search-query as well as changes on the selected attributes cause an asynchronous update of this table. So this table always contains an up to date result of the search, that cannot only be used to select specific techniques and attributes for the export, but also to browse through the database.

Similar again to the traditional solution, at the bottom of the result-table there can be a stylesheet selected that shall be used for export the actual table. What's different again is that the result of the export is not shown in a new page, but is attached to the actual page.

6 Reasons for choosing Ajax

From the technical point of view, we do not expect any improvements by using the Ajax implementation approach. Especially, we do not expect to create fewer or faster code. On the other hand, we do not expect a deterioration of performance as well. As Ajax mainly is a method for creating user interfaces, our decision is based on what we think gives the user a better tool for working with the data stored by the application. And, as laid out in section 5, we believe that the user can have a real benefit of an interface built using Ajax. We think, that the benefit can be outlined as:

- having a more clearly layout, consistent interface that provides a single view of the data, which the user can learn and use more quickly (the user gets faster what s/he wants from the application).
- the possibilities of DHTML and XMLHttpRequest enable the user to continue work, and save data to or request data from the server at the same time, so that the user does not lose the plot of what he's actually doing
- the interface is easier to handle

Because of these advantages, and from the fact that we do not see any disadvantage or problem, we suggest that it is appropriate to use the Ajax approach for implementation. As an alternative to Ajax, we could have picked out single specific technologies from the concept of Ajax, like using DHTML only or the XMLHttpRequest only. But as we laid out in chapter 5, both DHTML and XMLHttpRequest include benefits for the user, that would be lost if not both were used together. The use of (X)HTML and CSS is state-of-the-art for creating any kind of website or web-application, and therefore we did not discuss their use. One reason for using XHTML instead of HTML is that it is possible to simply hook in complete parts of the page received from the server to the DOM tree. The fact, that the ECMAScript provides a practical set of functions for processing XML, and that one of the core-functionality of the application is the transformation of XML data makes XML the logical joice for the transport format. And because of this, we decided to use the complete Ajax package.

7 Conclusion

In this paper, we discussed the capability of the Ajax design approach for the implementation of a web application in a practical course, that helps to manage information about techniques for information vizualization. We presented Ajax, a suggestion for how to use ECMAScript, DHTML, XMLHttpRequest and X(HT)ML together to create user interfaces for web application that behave almost like desktop applications. Before our decision for or against the use of Ajax, we summarized advantages and disadvantages of Ajax. The most important of them are listes in table 2. We also summarized suggestions on how

advantages	*disadvantages*
- Possibility to create ”rich internet applications” (=improved usability) - no plugins required - based on standards	- problems with browser functions (”back button”, bookmarking) - development time increases - not all clients support Ajax (although most)

Table 2: The most important advantages and disadvantages of Ajax

to use Ajax and what should be avoided. We then compared these suggestions with the needs of the specific application, and how we implement them, or why we ignore them. For arriving at a decision about the use of Ajax, we compare the feasible user interfaces with Ajax against the traditional approach. We concluded, that the interface created with Ajax enables the user to work more efficiently with the application. We therefore came to the conclusion to use Ajax for the implementation of the specific application.

8 References

References

[Garret,2005a] Jesse J. Garrett, Ajax: A New Approach to Web Applications. Adaptivepath. Created at: February 18, 2005. Retrieved at: November 27, 2005. http://www.adaptivepath.com/publications/essays/archives/000385.php

[Garret,2005b] Jesse J. Garrett, Why Ajax Matters Now. Created at: Sempember 16, 2005. Retrieved at: November 26, 2005. http://www.ok-cancel.com/archives/article/2005/09/why-ajax-matters-now.html

[Willison,2005] Simon Willisons, Why the term Ajax is useful. Created at April 19, 2005. Retrieved at: November 27, 2005. http://simon.incutio.com/archive/2005/04/19/useful

[Rees,2002] Michael J. Rees. Evolving the browser towards a standard User Interface Architecture. In *ACM International Conference Proceeding Series; Vol. 20 - Third Australasian conference on User interfaces*, pages 1-7, Melbourne, Victoria, Australia, 2002. ACM.

[Walton et al.,2000] Joan D. Walton, Robert E. Filman, Dadiv J. Korsmeyer. The evolution of the DARWIN system. In textitProceedings of the 2000 ACM symposium on Applied computing - Volume 2, pages 971 - 977, Como, Italy, 2000. ACM.

[Fraternali,1999] Piero Fratenali. Tools and Approaches for Developing Data-Intensive Web Applications: A Survey. *ACM Computing Sirveys*, 31(3): 227-263, September 1999.

[Hegaret et al.,2005] Philippe Le Hegaret et al., DOM. W3C. Created at January 19, 2005. Retrieved at: November 27, 2005. http://www.w3.org/DOM/

[Hors et al.,2004] Arnaud Le Hors, Philippe Le Hegaret, Lauren Wood, Gavin Nicol, Jonathan Robie, Mike Champion, Steve Byrne: Document Object Model (DOM) Level 3 Core Specification. W3C. Created at April 7, 2004. Retrieved November 23, 2005. http://www.w3.org/TR/2004/REC-DOM-Level-3-Core-20040407/

[Champeon,2001] Steve Champeon, JavaScript: How Did We Get Here? O'Reilly. Created at: June 4, 2001. Retrieved at: November 24, 2005. http://www.oreillynet.com/pub/a/javascript/2001/04/06/js_history.html

[Hixie,2005] Ian Hixie, Hixie's Natural Log: Call an apple an apple. Created at: March 3, 2005. Retrieved at: November 27, 2005. http://ln.hixie.ch/?start=1111339822&count=1

[Schaefer,2005] Mathias Schaefer, Dynamisches HTML. SELFHTML. Created at: November 24, 2005. Retrieved at: November 28, 2005.

[O'Reilly,2005] Tim O'Reilly, What Is Web 2.0. O'Reilly. Created at: September 30, 2005. Retrieved at: November 26, 2005. http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html

[LaMonica,2005] Martin LaMonica, AJAX gives software a fresh look. cnet. Created at: October 4, 2005. Retrieved at: October 27, 2005. http://news.com.com/Ajax+gives+software+a+fresh+look/2100-1007_3-5886709.html

[Girensohn and Lee,1998] Andreas Grigensohn, Alison Lee. Developing Collaborative Applications On the World Wide Web. In *CHI 98 conference summary on Human factors in computing systems*, pages 141-142, Los Angeles, California, United States, April 1998. ACM

[Bryant,2005] Stephen Bryant, AJAX Powers Web 2.0 Growth. Created at: October 5, 2005. Retrieved at: October 24, 2005. http://www.publish.com/article2/0,1895,1867584,00.asp

[Google,2005a] Google, GMail. Google. Retrieved at: November 27, 2005. http://mail.google.com

[Google,2005b] Google, Google Map. Google. Retrieved at: November 27, 2005. http://maps.google.com

[Flickr,2005] Yahoo, flickr. Yahoo. Retrieved at: November 27, 2005. http://www.flickr.com/

[Obasanjo,2005a] Dare Obasanjo, SOA, AJAX and REST: The Software Industry Devolves into the Fashion Industry. Created at: March 22, 2005. Retrieved at: November 25, 2005. http://www.25hoursaday.com/weblog/PermaLink.aspx?guid=018ea507-4a62-4493-b01b-321e3672d725

[Papageorge,2005b] Mike Papageorge, On Ajax and Marketing Technologies. Created at: February 28, 2005. Retrieved at: November 26, 2005. http://www.fiftyfoureleven.com/weblog/news-rants-and-ephemera/on-ajax-and-marketing-technologies

[Haughey,2005] Matthew Haughey, Note to geeks: look beyond the end of your nose. Created: April 18, 2005. Retrieved: November 25, 2005. http://a.wholelottanothing.org/2005/04/note_to_geeks_1.html

[Ginda,2004] Robert Ginda, JavaScript. Mozilla. Created at: September 9, 2004. Retrieved at: November 24, 2005. http://www.mozilla.org/js/

[Wilkens,2002] Linda Wilkens. Objects with prototype-based mechanisms. *Journal of Computing Sciences in Colleges*, 17(3): 131-140, February 2002.

[ECMA,1999] Standard ECMA-262, third edition. ECMA. Created at: December 1999. Retrieved at: November 13, 2005. http://www.ecma-international.org/publications/standards/Ecma-262.htm

[Baron,2004] David Baron, Mozilla Layout Engine. Mozilla. Created at: September 15, 2004. Retrieved at: November 25, 2005. http://www.mozilla.org/newlayout/

[Ginda,2000] Robert Ginda, What is in JavaScript 1.5? Mozilla. Created at: December 31, 2004. Retrieved at: November 26, 2005. http://www.mozilla.org/js/js15.html

[Microsoft,2005a] Microsoft, JScript. Microsoft. Retrieved at: November 26, 2005. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/script56/html/js56jsoriJScript.asp

[Ward and Smith,1998] Robert Ward, Marthin Smith. JavaScript as a First Programming Language for Multimedia Students. In *Proceedings of the 6th annual conference on the teaching of computing and the 3rd annual conference on Integrating technology into computer science education: Changing the delivery of computer science education*, pages 249-253, Dublin, Ireland, August 1998. ACM Press.

[Microsoft,2005b] Microsoft, XMLHttpRequest. Microsoft. Retrieved at: November 26, 2005. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/xmlsdk/html/7924f6be-c035-411f-acd2-79de7a711b38.asp

[Zammetti,2005] Frank W. Zammetti, Ajax using XMLHttpRequest and Struts. Retrieved at: November, 27, 2005. http://www.omnytex.com/articles/xhrstruts/xhrstruts.pdf

[Apple,2005] Apple, Dynamic HTML and XML: The XMLHttpRequest Object. Apple. Created at: June 24, 2005. Retrieved at: November 27, 2005. http://developer.apple.com/internet/webcontent/xmlhttpreq.html

[Quin et al.,2005] Liam Quin et al., XML. W3C. Created at November 23, 2005. Retrieved at: November 27, 2005. http://www.w3.org/XML/

[Thompson et al.,2005] Henry Thompson, Philippe le Hegaret, XML Core Working Group Public Page. W3C. Created at: November 23, 2005. Retrieved at: November 27, 2005. http://www.w3.org/XML/Core/

[Yergeau et al.,2004a] Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, Eve Maler, Francois Yergeau, Extensible Markup Language (XML) 1.0 (Third Edition). W3C. Creted at February 4, 2004. Retrieved at: November 27, 2005. http://www.w3.org/TR/2004/REC-xml-20040204/

[Yergeau et al.,2004s] Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, Eve Maler, Francois Yergeau, John Cowan, Extensible Markup Language (XML) 1.1. W3C. Created at April 15, 2004. Retrieved at: November 27, 2005. http://www.w3.org/TR/2004/REC-xml11-20040204/

[Sundaresan and Moussa,2001] Neel Sundaresan, Reshad Moussa. Algorithms and Programming Models for Efficient Representation of XML for Internet Applications. In *Proceedings of the 10th international conference on World Wide Web*,pages 366-375, Hong Kong, Hong Kong, May 2001. ACM Press.

[W3C,2005] W3C, HyperText Markup Language (HTML). W3C. Created at: October13, 2005. Retrieved at: November 25, 2005. http://www.w3.org/MarkUp/

[Mahemoff,2005] Michael Mahemoff, Ajax Patterns. Created at: November 27, 2005. Retreived at: November 27, 2005. http://www.ajaxpatterns.org

[Bergmann and Bormann,2005] Olaf Bergmann, Carsten Bormann. *AJAX Frische Ansaetze fuer das Web-Design*. Teia Lehrbuch Verlag, Berlin, 2005.

[Fried,2005] Jason Fried, XMLHttpRequest, Ajax, and the customer experience. Created at: February 24, 2005. Retrieved at: November 26. 2005. http://www.37signals.com/svn/archives/001070.php

[Baekdal,2005] Thomas Baekdal, XMLHttpRequest Usability Guidelines. Created at February 20, 2005. Retrieved at November 26, 2005. http://www.baekdal.com/articles/Usability/XMLHttpRequest-guidelines/

[Bosworth,2005a] Alex Bosworth, Ajax Mistakes. Created at May 18, 2005. Retrieved at November 26, 2005. http://alexbosworth.backpackit.com/pub/67688

[Bosworth,2005b] Adam Bosworth, Ajax Reconsidered. Created at: June 1, 2005. Retrieved at: November 27, 2005. http://www.adambosworth.net/archives/000044.html

[Stenhouse,2005] Mike Stenhouse, Fixing the Back Button and Enabling Bookmarking for AJAX Apps. Created at: June 13, 2005. Retrieved at: November 27, 2005. http://www.contentwithstyle.co.uk/Articles/38/fixing-the-back-button-and-enabling-bookmarking-for-ajax-apps

[Lopp,2005] Michael Lopp, The Web Application Leap. Created at: May 24, 2005. Retrieved at: November 26, 2005. http://www.randsinrepose.com/archives/2005/05/24/the_web_application_leap.html

[Massy,2005] Dave Massy, Ajax==DHTML+XMLHttp. Created at March 20, 2005. Retrieved at: November 27, 2005. http://blogs.msdn.com/dmassy/archive/2005/03/20/399412.aspx

[Neuberg,2005] Brad Neuberg, AJAX: How to Handle Bookmarks and Back Buttons. O'Reilly. Created at: October 26, 2005. Retrieved at: November 26, 2005. http://www.onjava.com/pub/a/onjava/2005/10/26/ajax-handling-bookmarks-and-back-button.html

[Austin,2005] Austin, Best Practices: Implementing javascript for rich internet applications. Created at: July 27, 2005. Retrieved at: November 27, 2005. http://thinkingandmaking.com/entries/63

[Campbell,2005] Ryan Campell, The Hows and Whys of Degradable Ajax. Created at: September 6, 2005. Retrieved at: November 27, 2005. http://particletree.com/features/the-hows-and-whys-of-degradable-ajax/

[Grossman,2005] Dan Grossman, What's Wrong With Ajax. Created at: September 21, 2005. Retrieved at: November 27, 2005. http://www.aventureforth.com/2005/09/21/whats-wrong-with-ajax/

[Almaer,2005] Dion Almaer, Java Web Server: Jetty 6.0 Continuations for Ajax Architectures. Created at: September 17, 2005. Retrieved at: November 27, 2005. http://ajaxian.com/archives/2005/09/java_web_server.html

[Mason et al.,2001] James D. Mason et al., ISO 8879:1986 – SGML. ISO. Created at: August 13, 2001. Retreved at: November 27, 2005. http://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail
?CSNUMBER=16387&ICS1=35&ICS2=240&ICS3=30