JavaScript Frameworks and Ajax Applications

Adam Domański[1], Joanna Domańska[2], and Sebastian Chmiel[1]

[1] Institute of Informatics,
Silesian Technical University,
Akademicka 16, 44-100 Gliwice, Poland
adamd@polsl.pl
[2] Institute of Theoretical and Applied Informatics
Polish Academy of Sciences
Baltycka 5, 44-100 Gliwice, Poland
joanna@iitis.gliwice.pl

Abstract. JavaScript frameworks are useful in the development of interactive web pages. Ajax technology supports a dialog with the WWW server. This article compares performance and functionality of the Ajax libraries for major Web browsers. The size of the libraries, their load time and execution time is compared. The article also evaluates the Document Object Model (DOM) support provided by selected libraries.

Keywords: Ajax, DOM, JavaScript, web applications.

1 Introduction

Ajax technology is an asynchronous communication between the browser and the server [1, 2]. This technology is currently standard in dynamic websites. It is used by such companies as Google, Facebook, YouTube and many others. This solutions improves the usability of web applications. Ajax gives the programmer more capabilities and allows to create applications with the same level of complexity as a normal window-based programs [3–6]. This techology becomes more and more popular and lot of new frameworks have been created.

The purpose of this paper is to compare the most popular Javascript software libraries supporting the Ajax technology. The article presents a functionality and a performance of the most popular frameworks. During the test such indicators as: size of library, loading time, execution time of the Ajax requests and event handling were considered.

This paper shows how different libraries behave in the specific cases and how to select the library to meet the programmer needs.

This paper is organized as follows. Section 2 gives a brief description of the Ajax technology and presents some popular Javascript libraries. Section 3 describes the experiments and gives some numerical results. Section 4 concludes this article.

A. Kwiecień, P. Gaj, and P. Stera (Eds.): CN 2014, CCIS 431, pp. 57–68, 2014.

2 Ajax Technology

In this section the Ajax (Asynchronous JavaScript And XML) technology and some popular Javascript libraries were briefly presented.

Ajax is not a new technology. It is a combination of several existing solutions [7–9]. The main idea of this technology is a completely new approach of communication between browser and server [10]. This technology provides a mechanism which allows to download new web page content using JavaScript without reloading the WWW page. In the standard solution, the browser sends the request to the server and the server sends another page in response [11]. Figure 1 presents such situation. Downloading the whole pages is sometimes unnecessary and redundant. In some situations we need to load only a small part of the page (i.e. the panel side management). In the case of Ajax technology it is possible to send asynchronous request to download only interesting content from the server [11–13].

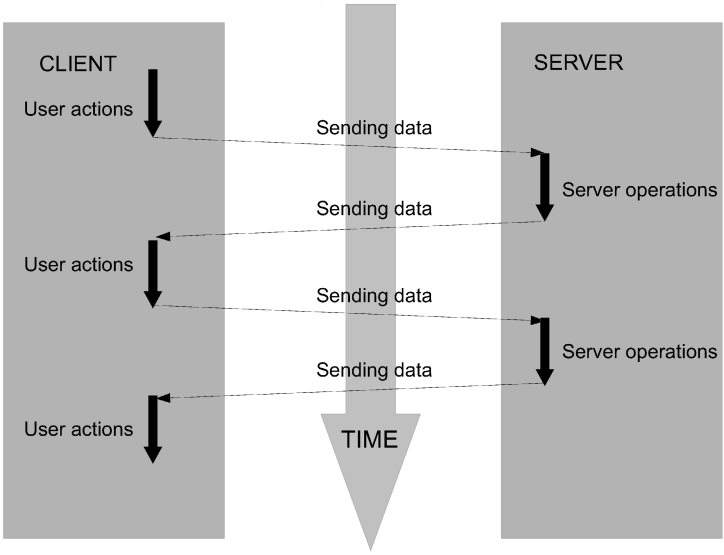

Fig. 1. Scheme of synchronous communication between the client and the web server

Figure 2 presents the asynchronus comunication between browser and server. The user event started the new content downloading procedure using JavaScript. The Ajax engine sends a request to the server (the web page has not been reloaded) and waits for new content (in background). After successful new content downloading, Ajax engine processes collected data and publish them on the site.

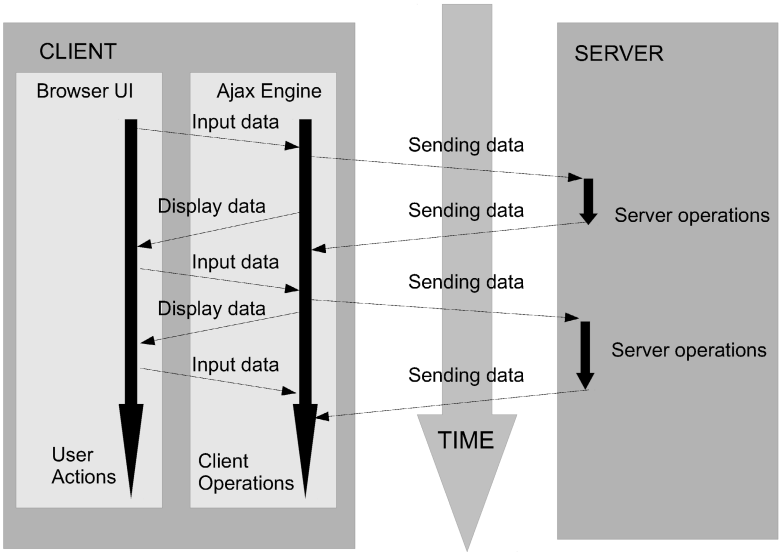

Fig. 2. Asynchronous requests to the server

An asynchronous communication with the server uses a special object called *XMLHttpRequest* [8]. Originally this object had a different name. It was created with Internet Explorer 5.0 and defined as an *ActiveX* controler [11]. Because other browsers began to use the similar object and called it the *XMLHttpRequest* from IE 7.0 the controler *ActiveX* was deprecated.

Ajax technology uses four data formats to exchange data with the server [14]:

- XML (eXtensible Markup Language),
- plain text,
- URL – data are transmitted in the URL (eg. *file.php?var1=value1&var2=value2*),
- JSON (JavaScript Object Notation).

Ajax can be applied in many solutions. Some of them are considered as essential of the Web 2.0 Internet [15], some are an unnecessary additions. The Ajax technology is suitable for presented below mechanisms:

- forms validations,
- autocomplete and word completion,
- edit in place – clicking on an element causes the appearance of the edit box with the ability to save the new value,
- backlight changes – suitable for services, which update the content from time to time without refreshing the page,
- slow loading – content from the top of the page are downloaded and shown at once.

There are many Ajax technology libraries. All of them provide basic functionality of managing asynchronous connection between browser and server, but generally they provide much greater opportunities. The most spectacular are: drag and drop, edit in place, data validation supporting, possibility of using defined widgets and templates, various widget skins, animations and visual effects [16–25]. Figure 3 presents the the functionality and compatibility of different libraries with selected browsers.

	DHTMLX	Dojo	Enyo	Ext JS	jQuery	MochiKit	MooTools	Prototype	Script.aculo.us	Yahoo UI
Browser support	IE 6+ Firefox 3+ Safari 4+ Opera 10.5+ Chrome 3+	IE 6+ Firefox 3.6+ Safari 5+ Opera 10.5+ Chrome 13+	IE 8+ Firefox 4+ Safari 5+ Opera 10+ Chrome 10+	IE 6+ Firefox 1.5+ Safari 3+ Opera 9+ Chrome 3+	IE 9+ Firefox 2+ Safari 3+ Opera 9+ Chrome 1+	IE 6+ Firefox 2+ Safari 2+ Opera 8.5+ Chrome 10+	IE 6+ Firefox 2+ Safari 3+ Opera 9+ Chrome 1+	IE 6+ Firefox 1.5+ Safari 2.0.4+ Opera 9.25+ Chrome 1+	IE 6+ Firefox 1.5+ Safari 3+ Opera 10+ Chrome 1+	IE 6+ Firefox 3+ Safari 4+ Opera 10+ Chrome 10+
DOM	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
JSON Format	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
AJAX	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
Button backward	✘	✔	✔	✔	✔	✔	✔	✔	✔	✔
Simple visual effects	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
Adanced visual effects	✘	✔	✔	✔	✔	✔	✔	✘	✔	✔
Additional widgets	✔	✔	✔	✔	✔	✔	✔	✘	✔	✔
Drag and drop	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
Edition in place	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
Forms validation	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
Pages templates	✔	✔	✔	✔	✔	✘	✔	✘	✘	✔

Fig. 3. Comparison of the functionality of selected libraries based on information from manufacturers websites

3 Performance of the Ajax Libraries

In this section the comparing of the performance of selected libraries were carried out. Durring the tests we checked:

- sizes of libraries in the basic version (Ajax-enabled DOM events and parsing the data) and in the expanded versions (graphics effect, forms and data validation),
- libraries load times in primary and extended versions,
- an execution time of the XMLHttpRequest method ("GET" or "POST"). This test was performed in the case of three typical feedback formats: plain text, XML, and JSON,
- searching and editing elements in the DOM model.

Each of described bellow tests (except library size checking) was conducted for five types of browser:

- Internet Explorer 10.0.9200,
- Firefox 23.0.1,
- Safari 5.1.7,
- Opera 12.16,
- Chrome 29.0.1547.

3.1 Library Size

We analysed the empty web page size with the loaded library. We considered: library with basic modules (functions responsible for the asynchronous communication with the server, DOM handling functions, events handling and parsing data), library with extended version (graphics effects handling and forms validation). The page size was measured using the Firefox plugin called "Firebug".

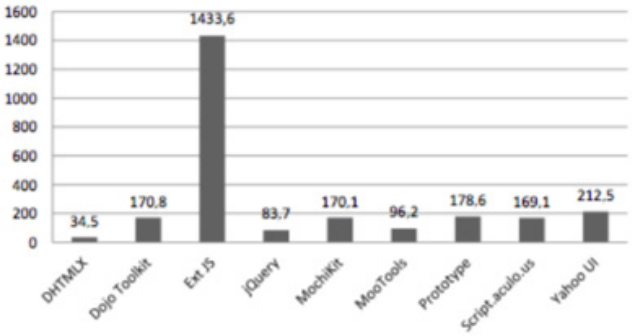

Fig. 4. Libraries sizes (basic versions) [KB]

We concluded that the worst case was Ext JS library. This is due to number of additional mechanisms and widgets builded into the base library file. The library authors did not divide modules, so it is necessary to load the entire library to handle basic functionality. Better results were obtained for the DHTMLX, jQuery and MooTools libraries. This is due to code optimization (unnecessary whitespace were removed, variable names have been changed and comments have been removed). Such optimization can save 80 % of the initial size of the code.

3.2 Library Load Time

We have measured the library loading time. The code from the previous test was used. The time was measured using a JS functions.

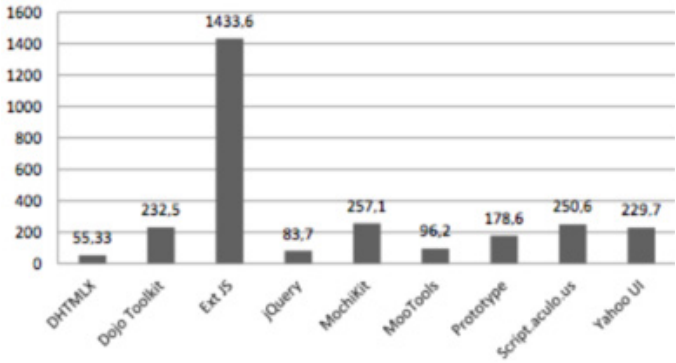

Fig. 5. Libraries size (extended versions) [KB]

Table 1. Library loading times (basic versions) [ms]

	IE	Firefox	Safari	Opera	Chrome	Avg.
DHTMLX	16	18	4	3	3	8.8
Dojo Toolkit	26	45	28	27	19	29
Ext JS	198	292	183	184	174	206.2
jQuery	20	29	14	16	11	18
MochiKit	49	35	23	21	11	27.8
MooTools	46	45	17	22	20	30
Prototype	26	45	17	22	20	24.8
Script.aculo.us	39	39	23	21	26	29.6
YahooUI	59	87	47	97	32	64.4

The results (Tables 1, 2) are similar to the results obtained in the previous test. This is due to the fact that the page loading time depends on its size. The difference in the time of loading of libraries in different browsers is not always the fault of the browser. Often this is due to the fact that the JavaScript language has worked differently in different browsers (usually older ones), and the authors of libraries provided a proper work in all browsers.

3.3 Sending and Receiving Ajax Request

In this subsection the most important mechanism of the Ajax technology were rated. There were no delays on the line during the client-server data transfer because the tests were carried out on the local server. Figure 6 presents the code of the experiment.

For each Ajax task the method for sending data has to be specified (GET or POST). The GET method sends the variables and their values in the page URL. The POST method sends variables in the header of HTTP. The experiment allowed to specify the transmitted data format (plain text, XML, JSON). The file content of all free formats is the same. This aproach allows to compare different

Table 2. Library loading times (extended versions) [ms]

	IE	Firefox	Safari	Opera	Chrome	Avg.
DHTMLX	4	22	9	4	3	8.4
Dojo Toolkit	33	47	31	53	23	37.4
Ext JS	198	292	183	184	174	206.2
jQuery	20	29	14	16	11	18
MochiKit	66	38	28	29	15	35.2
MooTools	46	45	17	22	20	30
Prototype	26	37	20	23	18	24.8
Script.aculo.us	76	58	33	47	32	49.2
YahooUI	56	99	49	87	32	64.6

formats for the same content. Text was endorsed with HTML tags (to be ready to display on the page). Its size was 4 125 bytes. In the case of the XML the file size was 4 157 bytes. Although the XML procesing is very simple but its additional tags occupies more space than other formats. The last variant was the JSON data format. The advantage of this format is a small data size (only 4 002 bytes) and the fact that this format is directly imported as an object in JavaScript. The common results are presented in Table 3 and Table 4.

The results of this test show that the data format and the transfer method do not play a significant role in the length of transfer time. The data processing is also fast. The differences between formats are insignificantly small. In most cases, one can notice better results for JSON (the size of the transmitted content is smallest). The fastest libraries in this case were DHTMLX, Dojo Toolkit and MooTools. The obtained times were less than 10 milliseconds.

3.4 Document Object Model Support

The data after downloading from the server are usually processed and displayed on the website. The object model of the web page called the DOM is used for webpage modifications. This subsection shows the performance of searching and editing elements in this model. Tested library mostly have their own mechanisms for handling DOM model. We evaluated the three most frequently performed operations:

- searching of elements (by different classifiers),
- modification of elements content,
- creating of new elements.

The single test was repeated 100 000 times due to short time of single test. Some libraries did not have its own mechanism for handling DOM. The tests for them are omitted and the values are indicated by 0.

For first case (searching the elements according to their attributes “id”), most libraries had short seek time. The worst of the tested libraries were MooTools

Table 3. Ajax request (method GET) [ms]

		IE	Firefox	Safari	Opera	Chrome	Avg.
DHTMLX	Plain text	2	11	4	4	10	6.2
	XML	1	17	4	9	8	7.8
	JSON	10	15	6	7	10	9.6
Dojo Toolkit	Plain text	3	9	7	4	5	5.6
	XML	4	6	6	5	5	5.2
	JSON	17	6	7	4	6	8
Ext JS	Plain text	6	35	14	6	42	20.6
	XML	47	39	14	6	45	22.2
	JSON	7	35	15	6	43	21.2
jQuery	Plain text	14	12	8	4	12	10
	XML	15	18	10	8	12	12.6
	JSON	14	17	10	5	11	11.4
MochiKit	Plain text	9	19	11	14	19	14.4
	XML	16	16	10	8	12	12.6
	JSON	16	16	10	12	18	14.4
MooTools	Plain text	13	9	11	4	10	9.4
	XML	13	9	9	5	12	9.6
	JSON	12	8	9	4	10	8.6
Prototype	Plain text	14	21	9	4	15	11.2
	XML	15	16	9	7	19	13.2
	JSON	14	12	9	4	18	11.4
Script.aculo.us	Plain text	15	20	9	5	13	12.4
	XML	19	13	10	7	13	12.4
	JSON	17	9	8	6	12	10.4
YahooUI	Plain text	3	21	8	14	22	16.6
	XML	13	21	8	15	23	16
	JSON	12	21	8	15	23	15.8

Table 4. AJAX request (method POST) [ms]

		IE	Firefox	Safari	Opera	Chrome	Avg.
DHTMLX	Plain text	5	19	5	7	8	8.8
Dojo Toolkit	Plain text	5	5	7	6	6	5.8
Ext JS	Plain text	8	28	14	9	46	2.1
jQuery	Plain text	9	16	11	8	13	11.4
MochiKit	Plain text	10	17	7	10	17	12.2
MooTools	Plain text	8	8	7	8	12	8.6
Prototype	Plain text	9	13	9	9	16	8.4
Script.aculo.us	Plain text	8	15	11	9	14	11.4
YahooUI	Plain text	7	21	9	18	22	15.4

```
<! DOCTYPE html>
<html>
<head>
   <meta charset="utf-8">
   <title> [ LIBRARY NAME] - [TEST NAME] < / title>
< / head>
<body>
   <h1 id="test_name"> [ LIBRARY NAME] - [TEST NAME] < / h1 >
   <div id="result_div"> [ RESULTS ] < / div >
   <script> [TIME FUNCTION - Loading ] < / script >
   <script>
      var resultDiv = document.getElementById (" result_div ");
      [ START TIME ]
      var test = fTestTimeCounter ();
      test.start ();
      new Ajax.Request ( [URL ] , {

      method: [ POST / GET ]
         parameters: {
            [ SENDING PARAMETERS ]
         }
         [ REQUEST SUCCESS ]
         onSuccess : function ( data) {
            var response = [DATA PROCESSING TEXT / XML / JSON ]
            [TIME STOP]
            alert ( test.stop () + ' ms ');
            [ DISPLAYING RESULTS ]
            resultDiv.innerHTML = " <pre> " + response + " < / pre> " ;
         }
         [ REQUEST FAILURE ]
         onFailure : function () {
            alert (' Something went wrong ...');
         }
      } ) ;
   < / script >
< / body>
< / html >
```

Fig. 6. AJAX requests – sending and receiving

i Yahoo UI. This situation is presented in Fig. 7. The searching element using identifier is most popular in JavaScript hence for all libraries the seek time was short.

Figure 8 presents the results achieved during the modification of the DOM elements. The libraries: Prototype, Script.aculo.us, and Yahoo UI changed slowly the values of the DOM elements. Very probably they do not use the prepared JavaScript function. The worst result 458 ms corresponds to 0.0458 ms for a single modification.

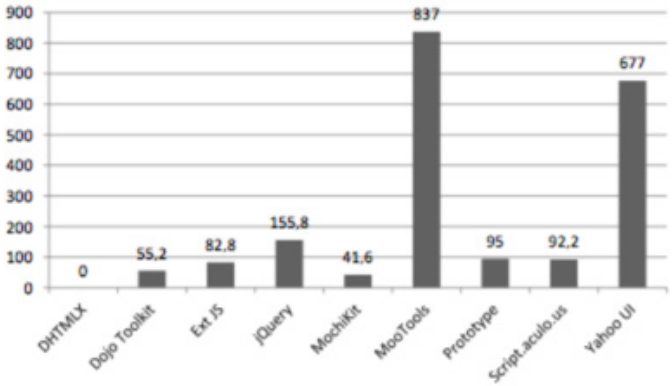

Fig. 7. Searching the element in the DOM according to the value attribute "id" [ms]

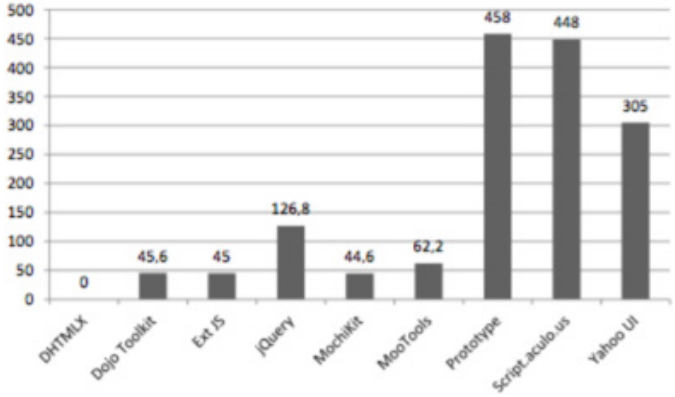

Fig. 8. Modification the element in the DOM [ms]

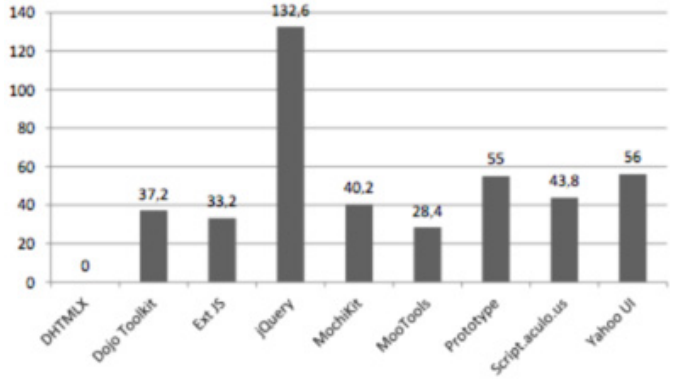

Fig. 9. Creating the element in the DOM [ms]

Creation of the DOM elements takes much more time than modifying their contents (Fig. 9). At the time of the creation two operations are performed: the creation of a new element and assign it to the specified parent. The obtained results are largely comparable. Only library JQuery works a few times slower (this time is highest in the case of Internet Explorer).

4 Conclusions

There is a large choice of available libraries and most of them have additional functionality. These additions can simplify the work programmer, but may be also an unnecessary ballast. Selected libraries can be divided into two categories: simple and complex. MochiKit and Prototype belong to the first type. They focus on the core functionality. The others hace additional modules with advanced operations: drag and drop functionality, advanced service forms with the data validation, edit in place and autocomplete. Most of them have the ability of creating visual effects.

The Open Source libraries were selected to tests. Their functionality were nearly identical hence the choice of appropriate library can be difficult. We assumed that the important factors are size and performance of library. Large library size negatively influence the page load time. The Ext JS library had the biggest size. However, it was characterized by good performance. The main element of Ajax speed communication with the server was alse examined. We obtained the best results for DHTMLX, dojo Toolkit and MooTools. In the case of transfer methods ("GET" and "POST") and data formats (plain text, XML, and JSON) performance is relatively similar. Small acceleration was noted in the case of the JSON. fIn this case the size of transfering data was the smallest.

The DOM operating tests confirmed the biggest differences between the studied libraries. Although single call lasted only milliseconds, this functionality may have great importance for large applications . The libraries: Dojo, MochiKit, Ext JS, Prototype, and Script.aculo.us were the fastest in the case of the test of seraching the structure of the DOM by identifier. Based on the test results the weaknesses of libraries Prototype, Script.aculo.us and Yahoo UI can be determined. The DOM elements modifying time was worst in the case of this libraries. In the case of inserting new elements the worst was jQuery library.

During our researches we noticed the differences in the quality of the available documentation. Some libraries had a well written tutorials with examples (Dojo Toolkit, jQuery, MooTools, Prototype). The other forced to longer search. Considering all the experiments, the best choice is the library Dojo Toolkit. Slightly worse are the prototype, Script.aculo.us, Ext JS and jQuery. and at the very end were MooTools and Yahoo UI. The worst results were obtained for MooTools and Yahoo UI.

References

1. Duda, C., Frey, G., Kossmann, D., Matter, R., Zhou, C.: AJAX Crawl: Making AJAX Applications Searchable. In: IEEE 25th International Conference on Data Engineering, ICDE (2009)

2. Tian, X.: Extracting Structured Data from Ajax Site. In: 2009 First International Workshop on Database Technology and Applications (2009)
3. Han, W., Di, L., Zhao, P., Li, X.: Using Ajax for desktop-like geospatial web application development. In: 2009 17th International Conference on Geoinformatics (2009)
4. Zepeda, J., Chapa, S.: From Desktop Applications Towards Ajax Web Applications. In: 4th International Conference on Electrical and Electronics Engineering, ICEEE (2007)
5. Weiguo, H., Liping, D., Peisheng, Z., Xiaoyan, L.: Using Ajax for desktop-like geospatial web application development. In: 2009 17th International Conference Geoinformatics (2009)
6. Dong, S., Cheng, C., Zhou, Y.: Research on AJAX technology application in web development. In: 2011 International Conference E-Business and E -Government, ICEE (2011)
7. Bruce, P.W.: Ajax Hacks Tips and Tools for Creating Responsive Web Sites. O'Reilly Media (2006)
8. Lauriat, S.M.: Advanced Ajax: Architecture and Best Practices. Prentice Hall (2008)
9. Lin, Z., Wu, J., Zhang, Q., Zhou, H.: Research on Web Applications Using Ajax New Technologies. In: International Conference on MultiMedia and Information Technology, MMIT 2008 (2008)
10. Jingjing, L., Chunlin, P.: jQuery-based Ajax general interactive architecture. In: 2012 IEEE 3rd International Conference on Software Engineering and Service Science, ICSESS (2012)
11. Vossen, G., Hagemann, S.: Unleashing Web 2.0: From Concepts to Creativity. Morgan Kaupmman (2007)
12. Matthijssen, N., Zaidman, A., Storey, M., Bull, I., van Deursen, A.: Connecting Traces: Understanding Client-Server Interactions in Ajax Applications. In: 2010 IEEE 18th International Conference on Program Comprehension, ICPC (2010)
13. Matthijssen, N., Zaidman, A.: FireDetective: understanding ajax client/server interactions. In: 2011 33rd International Confer. Software Engineering, ICSE (2011)
14. Zhang, X., Wang, H.: AJAX Crawling Scheme Based on Document Object Model. In: 2012 Fourth International Conference on Computational and Information Sciences, ICCIS (2012)
15. Al-Tameem, A.B., Chittikala, P., Pichappan, P.: A study of AJAX vulnerability in Web 2.0 applications. In: First International Conference on Applications of Digital Information and Web Technologies, ICADIWT 2008 (2008)
16. Powers, S.: Adding Ajax. O'Reilly Media (2007)
17. The DHTMLX library website, `http://dhtmlx.com/`
18. The Dojo Toolkit library website, `http://dojotoolkit.org/`
19. The Ext JS library website, `http://www.sencha.com/products/extjs`
20. The jQuery library website, `http://jquery.com/`
21. The library MochiKit website, `http://mochi.github.io/mochikit/`
22. The MooTools library website, `http://mootools.net/`
23. The Prototype library website, `http://prototypejs.org/`
24. The Script.aculo.us library website, `http://script.aculo.us/`
25. The Yahoo UI Library website, `http://yuilibrary.com/`