2017 8th International Conference on Information Technology (ICIT)

Evaluating IndexedDB Performance on Web Browsers

Ala'a Al-Shaikh, Azzam Sleit
Department of Computer Science
King Abdulla II School for Information Technology
University of Jordan
Amman, Jordan
alaamsh@hotmail.com, azzam.sleit@ju.edu.jo

Abstract— **HTML5 is the latest markup language that includes features to allow developers to do most of the production work throughout the browsers without relying too much on third-party add-ons. HTML5 is always accompanied with JavaScript that gives those browsers the ability to perform the program logic locally. IndexedDB is a NoSQL database supported by HTML5 that is capable of storing JSON objects. In this paper, we conducted a comparative analysis of IndexedDB performance on four renowned browsers, these are namely: Google Chrome, Mozilla Firefox, Internet Explorer, and Opera. IndexedDB operations are explored in each browser and a statistical analysis is conducted to compare the performance of these operations on each browser. Results show the lead for Internet Explorer with an overall performance of 96.43%, followed by Mozilla Firefox with an overall performance of 64.29%, then Opera with an overall performance of 57.14%, and finally Google Chrome with 32.14%.**

Keywords—IndexedDB; HTML5; Webpage Local Storage; Database Performance Analysis; Browser Databases; Client-Side Database

I. INTRODUCTION

Rich Internet Applications (RIAs) are of a great demand to users due to the diversity of functionality they provide [1]. Along with cloud computing and mobile applications, it became ubiquitous to Internet users [2]. The term Rich Internet Applications was first introduced by Macromedia [3]. An RIA transfers processing from being full-dependent on the server and specifies some parts of the web application to run on the client machine by means of the browser [4]. Offline web client (application) is a type of RIA, it supports users' operations without the need for the user to be connected to the Internet [5]. Offline web client (applications) leads us to the term Intermittently-Connected Web Applications (ICWA). An ICWA is a web application that reads and modifies data and keeps on functioning even though there is no connection to the Internet [6].

Hypertext Markup Language (HTML) is the language, or standard, used to develop web applications. According to W3C (the World Wide Web Consortium) "HTML is the standard markup language used to create web pages and its elements form the building blocks of all websites" [7]. HTML, HTML 2.0, HTML +, HTML 3.0, HTML 4, and HTML 5 are all versions of HTML. However, the first version of HTML that is used as a standard was HTML4 and hence HTML5. HTML5 comprises a set of new features such as: (1) native support for audio and video playback, which enables audio and video to be played by the browser itself without the need to install additional plugins, (2) file management and local storage, (3) client-side databases and offline caching, (4) geolocation features, and (5) it adds some new semantic tags to give meaning to building blocks of documents rather than concentrating on the design and representation [8].

Along with HTML, JavaScript is always present. JavaScript accompanies HTML development of webpages to make them dynamic. JavaScript is classified as a Dynamic Language, e.g. runtime code generation and dynamic object behavior [9], that gives developers the freedom to write code without any type information or typing disciplines [10]. JavaScript is used in several application areas in web applications such as: (1) tracking statistics, (2) interface enhancements, and (3) social integration [11]. In its basic form, HTML does not have the computational power to do arithmetic and logic, it also does not have the ability to access disk files, interpret user inputs, and access databases. JavaScript is used along HTML to provide it with the necessary power to do different types of operations and perform a wide range of tasks.

Because of the widespread of JavaScript and its popularity, it started to gain attention in the research field, especially in the areas of security, correctness, and performance [12]. In this paper, our main concern is the performance analysis of accessing browser databases, *IndexedDB* specifically.

IndexedDB is the database equipped with HTML5 and supported by some browsers. It is a low-level API that allows client-side storage of data. IndexedDB is a W3C recommendation, it performs data storage based on the key-

978-1-5090-6332-1/17/$31.00 ©2017 IEEE

value model. At the lowest level of detail, IndexedDB uses B Trees as a means of storing the data in the file system [13].

IndexedDB is a NoSQL database that follows the Key-Value model where records can be located using either a key or an index. Additionally, IndexedDB is supported by many browsers with no need to install additional plugins [6].

The remainder of this paper is organized as follows: in section II, a work related to this paper is illustrated, then in section III, the methodology is discussed. Some background about HTML5 and Java Object Notation (JSON) is given in section IV, and in section V IndexedDB is explored. Experimental Results are presented in section VI. Finally, in section VII conclusion and future work are highlighted.

II. Related Work

Kimak, S. et al. in their paper entitled "Performance Testing and Comparison of Client Side Databases Versus Server Side" [14], presented a performance testing of client side databases, and compared their work to server-side databases. They studied the performance of IndexedDB on two browsers: Firefox and Chrome. They concluded that the performance of IndexedDB on Firefox is better than its performance on Chrome. They also compared their results with a MySQL database, and their results show that client-side databases offer a good solution if compared to server-side databases.

Although Firefox and Chrome are the two widely-spread Internet browsers but there are still some browsers that were not addressed in their work. Internet Explorer is one of the most well-known and prominent browsers that most MS-Windows users are familiar with. It is included in all versions of Windows either for PCs or mobile devices. Opera is another browser that is taking its place between Internet users. In this paper, we extend the work done by Kimak, S. et al. and compare between four browsers, these are namely: Google Chrome, Mozilla Firefox, Internet Explorer, and Opera. Unlike the work of Kimak, S. et al. we are not interested in this paper in comparing the performance of server-side databases; Client-side databases are not a replacement to server-side databases, on the contrary, they compliment their work so as to provide better performance for web applications. That is why we do not compare between client-side and server-side databases in this paper.

III. Background

A. HTML5

The most well-known feature about HTML5 is that it is device independent; a browser that supports HTML5 is the only requirement to run HTML5 applications [15]. One of the most important features of web-based applications is their ability to work offline. This has several advantages: (1) bandwidth conservation, (2) server-load reduction, (3) web-lagging reduction [16, 17], (4) offline browsing, and (5) faster loading [18]. Using this offline features, applications can work locally with full functionality without the need to access the Internet, except when accessing the Internet is crucial [19, 20]. Figure 1 shows a web-applications architecture on both client and server sides [21].

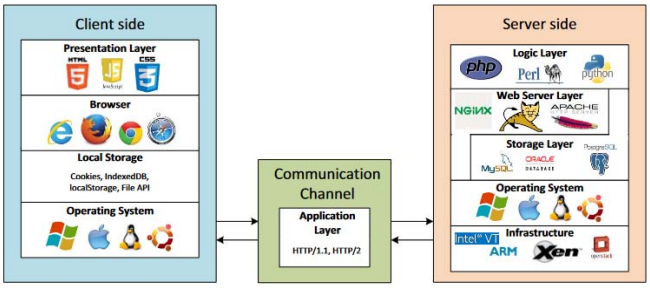

Fig. 1. Modern Web Application Architecture and Running Environments [20].

- Presentation Layer: consists of technologies for user interaction. These technologies include HTML for document markup and user input; JavaScript for scripting, programming, computations, and application logic; and CSS for formatting webpages.
- Browser Layer: which is an intermediate layer between the presentation and local storage. This layer may include HTML rendering engines, JavaScript interpreters, and XML parsers [21].
- Local Storage Layer: as shown in Fig. 1, this layer comprises cookies, IndexedDB, localStorage, and File API. Although, IndexedDB is dependent on the browser type, but it is closely related to the file system since the data is stored in files in the file system.
- Operating System Layer.

In this paper, we are only interested in the client side of a web application. Mainly, the first three layers are incorporated into this research. In the presentation layer HTML and JavaScript are used as the testing environment. In the second layer, the browser layer, a number of browsers are explored. IndexedDB is explored at the Local Storage layer. In the last layer we work only with one operating system, that is the Microsoft Windows operating system.

B. JavaScript Object Notation (JSON):

JavaScript Object Notation (JSON) is a data exchange format. It is similar to XML, but unlike XML it does not use tags, JSON uses text format, which makes parsing JSON objects, which can be looked at as string arrays, much faster and can be generated easier if compared to XML [22]. Although JSON was first introduced for data exchange, nowadays, it is being used by many programming languages to serialize and exchange internal objects [23].

One of the advantages of using JSON rather than XML, is that JSON is a text (string) representation that can be read by any programming language and parsed easily, while in the case of XML, there must be a parser to interpret XML.

Figure 2 shows a JSON example, in which three book objects, each contains some data about a certain book, and all of them are contained in the JSON object called books. The three objects form an array; they are enclosed between opening and closing square brackets ([and]) to denote the start and end of the array, while objects themselves are enclosed between opening and closing curly braces ({ and }). The name of the array in our example is books.

```
{"books":[
  {"title":"bk1","author":"at1","year":2016},
  {"title":"bk2","author":"at2","year":2013},
  {"title":"bk3","author":"at2","year":2002}
]}
```

Fig. 2. JSON object example.

IV. METHODOLOGY

In this paper, we compared the time required by each browser to execute some database operations, these are: (1) open database, (2) create object store, (3) close database, (4) create record, (5) read record, (6) update record, and (7) delete record. At the beginning, we generated a dataset that contains a number of JSON objects. Most of the operations to be tested use the objects stored in the dataset. We executed each operation on the four browsers several number of times and recorded the time elapsed to execute the operation.

V. INDEXEDDB

IndexedDB comprises many useful features. First, it is local, i.e. it resides on the client machine; not only is it local to the client's machine, it is also local to the application itself, in the sense it is local to the website that created it. So, IndexedDBs created by one website are not accessible by other websites, and therefore IndexedDBs created by the browser are not accessible to other programs even on the same machine.

Another important feature of IndexedDB is that it is a NoSQL (Not only SQL) database. The main properties of a relational database are ACID (Atomicity, Concurrency, Isolation, and Durability), in the case of NoSQL availability and scalability are of much concern, which led to the BASE properties which are: Basically Available, Soft state, and Eventually consistent [24, 25] (all access will return the same object if no updates are made to the object [26]). NoSQL databases are much concerned with availability and scalability, this demonstrates why they trade-off the ACID model for these properties [24, 27].

Unlike relational databases which has a common language of manipulation, namely the Structured Query Language (SQL), NoSQL has no common query language [28]. Each product has its own proprietary language to access the data, basically to perform the CRUD (Create, Read, Update, and Delete) operations [29].

NoSQL has different data models:

1) Key-Value Store: data is organized in objects, each has a unique key that identifies the record in the store. Key-Value is the simplest implementation amongst other types of NoSQL data models [30].

2) Document-Oriented Storage: similar to or based on the key-value store, in which a document is a key-value collection of attributes and values. Each document has a unique key that identifies the whole document [31, 30].

3) Graph Databases: store labelled nodes and edges to incorporate relationships between them [31]. Data that can be stored in this type may include maps, computer networks and social networks [30, 32].

4) Column Family Databases: they can be considered a database schema [31]. These are a mix between the key-value stores and the classical relational model (row/column structure) [30].

IndexedDB lies in the first type. It is a key-value store which stores objects of different structures, all of them are identified by a unique key that is used as an index to facilitate the retrieval of a specific object [14]. IndexedDB uses object stores [33].

IndexedDB is built on a transactional database model [33]. This is a powerful feature of IndexedDB. As long IndexedDB is a local database that runs in the browser. Most today browsers support multi-tabs; opening another instance of the database from another tab with the absence of transaction management could cause the application to crash.

Security is maintained by the same-origin policy [34]. The same-origin policy is a security model used to protect web applications [35]. In this model, cookies, JavaScript, and recently IndexedDB from once site are not allowed to interact with each other even though they all exist in the same browser on the same machine [36]. Databases are protected within the same-origin policy by controlling access to the same domain, application-layer protocol, and port [34].

Each browser has its own implementation of IndexedDB, e.g. Internet Explorer uses the Extensible Storage Engine, Firefox uses SQLite, and Chrome uses LevelDB [37].

IndexedDB is an object-oriented database rather than a relational database. Similar to the relational model where the majority of database operations are concentrated around tables, in IndexedDB the concentration is around the object store. Object stores are created and JavaScript objects are persisted to those stores. In the store objects are referred to as records; each record has a key and a value (key-value pair) and records are sorted according to the keys in an ascending order [34].

A key, which can be looked at as a primary key in relational databases, is always associated with a value which is a JSON object that has no predefined structure. The key can be either

automatically generated in an ordered sequence, or extracted from the value (the JSON object) based on the key path [34, 38].

In this paper, we classified the operations on IndexedDB into Basic Operations and CRUD Operations:

1) Basic Operations: these are the operations that do not directly affect the data, they affect the database itself, and they must be executed to ensure a proper working scenario. These operations include:

- Open Database: this is the first and most essential operation when working with databases. It incorporates establishing a connection to the database and returning an instance to the database object.
- Create Object Store: this operation is executed only once just when opening the database for the first time. If the database does not already exist, control moves to the *create object store routine* to create an object store. After the object store is created successfully, control moves back to the open database routine to open the requested database and return the requested database object.
- Close Database Connection: this is the last operation called to close the database connection. The connection is closed when all transactions are finished.

All of the previous operations are executed once; the open is executed at the beginning of the working session with the database, the same applies to create object store which only runs if the database has not been created yet, and the close operation is executed at the end of the database session.

2) CRUD Operations: as mentioned earlier, there is no common language, such as SQL, to support all NoSQL databases. Thus, each database has its own proprietary functions (methods) as we will see soon. All of the following CRUD operations are methods of the `IDBObjectStore` interface of the IndexedDB API [39].

- Create: the create operation is represented by and `add()` function. It creates new objects, ~~in,~~ i.e. add new objects, to the object store. If an object with the same key already exists in the object store, the operation will break and an error will be returned [40]. This operation is a counterpart to the `insert` operation in SQL language.
- Read: `get()` method is used to read data from the object store based on the key or index passed to the function as a parameter [41]. The renowned `select` statement is the counterpart for this function.
- Update: the SQL `update` statement is used to update the values of the records. Things are different in IndexedDB `put()` method; The same `put()` method can be used to add new records or update existing ones in an object store [42].
- Delete: The `delete()` method deletes the object whose key is passed to the method as a parameter [43]. It is similar to the SQL `delete` statement.

VI. Experimental Results

A. Testing Environment

All tests are conducted on an Intel Core(TM) i5-3230M CPU of 2.60 GHz and 3 MB cache with 4 cores. Memory size is 4 GB of RAM (2.87GB is only usable). The PC runs windows 7 Enterprise edition 64-bit.

Datasets are generated using a Java application and saved to disk as JSON objects. Each JSON object (record) comprises a number of fields; data in the fields are also generated randomly.

The browsers that have been chosen for the study are shown in Fig. 3 each along with its version.

Fig. 3. Browsers in the study.

Reasons of choosing these browsers are: (1) they all run on PCs, (2) they are very well-known and the most used browsers among Internet users, (3) they all support HTML5, (4) they all support IndexedDB, and (5) they all run under the Windows operating system.

Each operation is executed a sufficient number of times on all four browsers. Each time, *elapsed time* is recorded, that is the time required to execute the operation once. At the end, the average elapsed time for an operation is calculated.

B. Results:

For simplicity, abbreviations are used as follows to denote browsers: (1) GC: Google Chrome, (2) MF: Mozilla Firefox, (3) IE: Internet Explorer, and (4) OP: Opera.

Figure 4 summarizes the Open Database average times for different browsers showing the lead for the Opera browser.

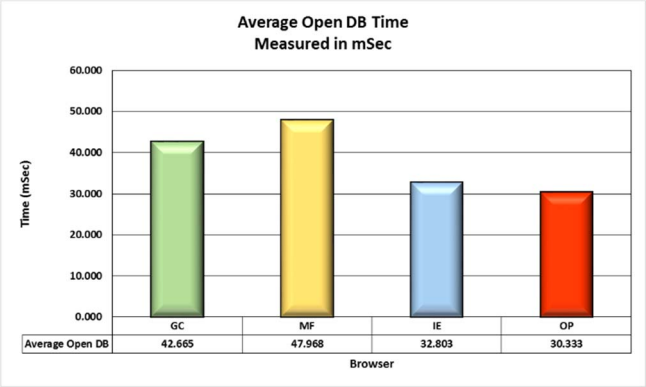

Fig. 4. Comparison between average open database times in milliseconds.

Figure 5 shows a comparison between average runtimes for Create Object Store and Close Database. IE then MF achieving the best results, while GC and OP were exchanging the last two ranks.

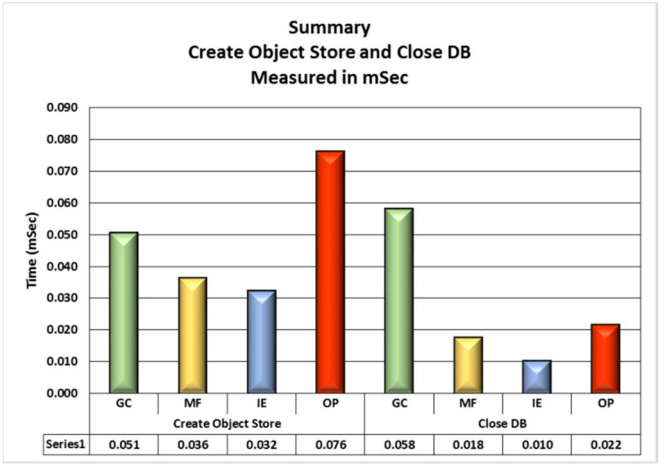

Fig. 5. Create object store and close database times in milliseconds.

Figure 6 combines the four CRUD operations together and compare between them. The x-axis of Fig. 6 is divided into four blocks, each is dedicated to one operation, and on each block average runtimes for each browser is shown. The y-axis represents the average runtime in milliseconds.

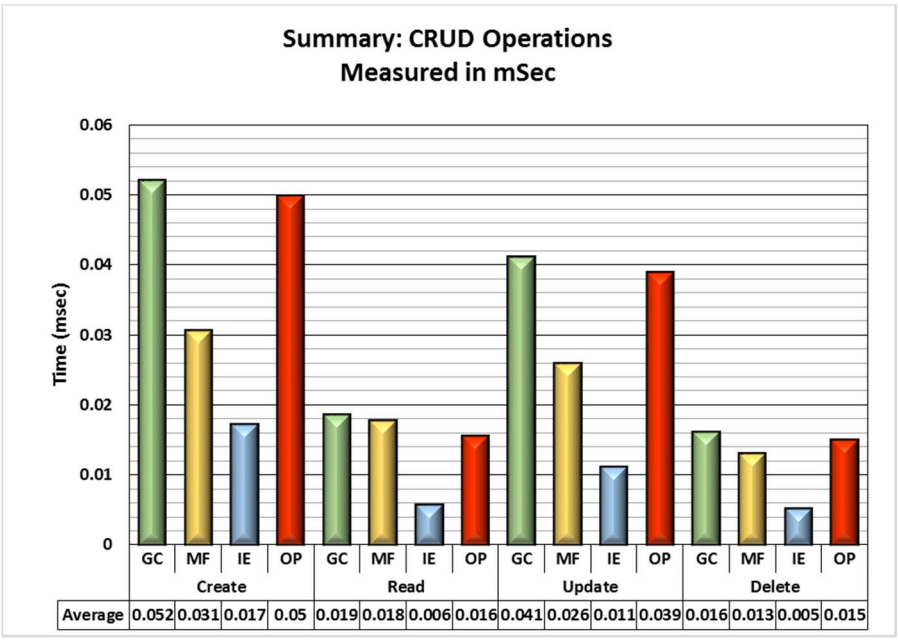

Fig. 6. Comparison between averages CRUD operations times measured in milliseconds.

Accordingly, IE shows the best performance over all other browsers. GC has the worst performance over all in almost all CRUD operations.

Now, to measure the overall performance of IndexedDB on each browser, we calculated the rank for each browser in each operation. Since we have four browsers, then we only have four ranks, the first weights 4, the second weights 3, the third weights 2, and the fourth and weights 1. For a total of seven operations experimented in this paper, the maximum score a browser can achieve is 28. We counted the number of times each browser fell in each of the four ranks, multiplied that count by the weight of the rank, this calculates the total score of each browser. Then the total score is divided by 28 as shown in TABLE I.

TABLE I. RANKING OF BROWSERS AND PERFORMANCE CALCULATION.

Browser	Rank				Total Score (28)	Overall Performance
	First (4)	Second (3)	Third (2)	Fourth (1)		
GC	0	0	2	5	9	32.14%
MF	0	5	1	1	18	64.29%
IE	6	1	0	0	27	96.43%
OP	1	1	4	1	16	57.14%

Figure 7 summarizes the overall performance of the four browsers, showing Internet Explorer in the lead, followed by Mozilla Firefox, then Opera, and finally Google Chrome.

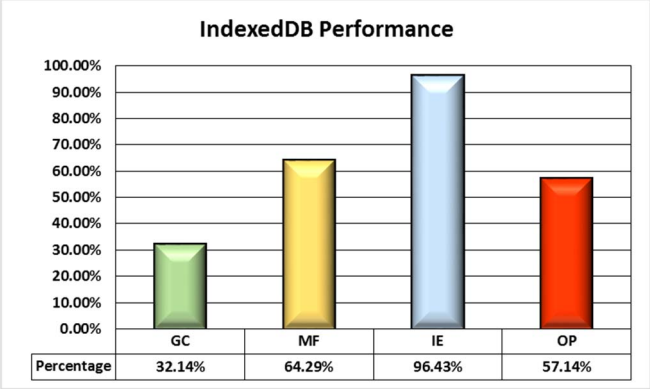

Fig. 7. Overall IndexedDB Performance on all browsers.

The main observation we get from the results is that IE came in the first place due to conducting this research under the MS-Windows operating system environment. IE is the web browser equipped originally with MS-Windows and it is natively designed to work under that environment. The remaining browsers are cross-platform and their performance might be platform dependent. We might obtain different results if the same research is conducted on different OS environments, keeping in mind that IE works only under MS-Windows, which dictates that it will not participate in any further investigations under different platforms.

Finally, we compared the results we obtained in this research to the ones in literature [14]. Only two browsers were incorporated in the experiments in literature [14], these were namely: Mozilla Firefox and Google Chrome. The results in the literature showed that IndexedDB performs faster in Firefox than Chrome [14]. This conforms to the results we obtained in this research which also show higher performance for Firefox than Chrome. Consequently, results of the two papers converge.

VII. CONCLUSION AND FUTURE WORK

In this paper, we presented a performance analysis for IndexedDB on four browsers: Google Chrome, Mozilla Firefox, Internet Explorer and Opera. Seven operations were tested that are classified into two sets of operations: Basic Operations which include Open Database, Create Object Store, and Close Database. The second set of operations is the CRUD Operations which include Create, Read, Update, and Delete records. Results show the lead for Internet Explorer with an overall performance of 96.43%, followed by Mozilla Firefox with an overall performance of 64.29%, then Opera with an overall performance of 57.14%, and finally Google Chrome with 32.14%.

Further tests can be conducted on several other operations of the IndexedDB as a future work. Also, the performance can be measured on different operating systems such as Linux, or on mobile devices with different operating systems, such as Android or iOS. Another area of future work can be the comparison between the performance of IndexedDB and SQLite which is used in most mobile devices.

REFERENCES

[1] L. Philips, W. D. Meuter and C. D. Roover, "Poster: Tierless Programming in JavaScript," in 2015 IEEE/ACM 37th IEEE International Conference on Software Engineering (ICSE), 2015 .

[2] A. Al-Shaikh, H. Khattab, A. Sharieh and A. Sleit, "Resource Utilization in Cloud Computing as an Optimization Problem," *International Journal of Advanced Computer Science and Applications (IJACSA),* vol. 7, no. 6, pp. 336-342, 2016.

[3] P. Dolog and J. Stage, "Designing Interaction Spaces for Rich Internet Applications with UML," in 7th International Conference, ICWE 2007 Como, Italy, July 16-20, 2007 Proceedings, Springer Berlin Heidelberg, 2007, pp. 358-363.

[4] D. Woolston, "Rich Internet Applications," in Pro Ajax and the .NET 2.0 Platform, Apress, 2006, pp. 91-100.

[5] J. Song, G. Yu, D. Wang and T. Nie, "Offline Web Client: Approach, Design and Implementation Based on Web System," in Web Information Systems – WISE 2006, Springer Berlin Heidelberg, 2006, pp. 308-314.

[6] Leff, J. T. Rayfield, R. Konuru and R. Balasubramanian, "Untether: Middleware Components to Support Intermittently Connected Web Applications," in 2013 IEEE 20th International Conference on Web Services, 2013 .

[7] W3C, "W3C HTML," W3C, [Online]. Available: https://www.w3.org/html/. [Accessed 29 April 2016.[

[8] "HTML5: A New Standard for the Web," Medical Reference Services Quarterly, vol. 30, no. 1, pp. 50-55, 2011 .

[9] S. Wei and B. G. Ryder, "Taming the dynamic behavior of JavaScript," in SPLASH '14 Proceedings of the companion publication of the 2014 ACM SIGPLAN conference on Systems, Programming, and Applications: Software for Humanity, 2014 .

[10] M. Pradel, P. Schuh and K. Sen, "TypeDevil: Dynamic Type Inconsistency Analysis for JavaScript," in 2015 IEEE/ACM 37th IEEE International Conference on Software Engineering (ICSE), 2015 .

[11] N. Nikiforakis, L. Invernizzi, A. Kapravelos, S. V. Acker, W. Joosen, C. Kruegel, F. Piessens and G. Vigna, "You Are What You Include: Large-scale Evaluation of Remote JavaScript Inclusions," in CCS '12 Proceedings of the 2012 ACM conference on Computer and communications security, 2012 .

[12] G. Richards, S. Lebresne, B. Burg and J. Vitek, "An Analysis of the Dynamic Behavior of JavaScript Programs," in PLDI '10 Proceedings of the 31st ACM SIGPLAN Conference on Programming Language Design and Implementation, 2010 .

[13] W3C, "Indexed Database API," W3C, [Online]. Available: https://www.w3.org/TR/IndexedDB/. [Accessed 29 April 2016.[

[14] S. Kimak and J. Ellman, "Performance Testing and Comparison of Client Side Databases Versus Server Side," in 14th Annual Post-Graduate Symposium on the Convergence of the Telecommunications, Networking and Broadcasting (PGnet 2013), 2013 .

[15] Intermec, "Does HTML5 Make Sense for Mobile Enterprise Applications?," Honeywell International Inc., 2013.

[16] Z. Tahir, T. Inamoto, Y. Higami and S. Kobayashi, "The Analysis of Automated HTML5 Offline Services (AHOS)," in 2015 International Conference on Intelligent Informatics and BioMedical Sciences (ICIIBMS), 2015 .

[17] H. Al-Hasan, M. Qatawneh, A. Sleit and W. Almobaideen, "EAPHRN: Energy-Aware PEGASIS-Based Hierarchal Routing Protocol for Wireless Sensor Networks," Journal of American Science, vol. 7, no. 8, pp. 753-758, 2011 .

[18] H. P. D. Purnamasari and N. Syifana, "Clickable and interactive video system using HTML5," in The International Conference on Information Networking 2014 (ICOIN2014), 2014 .

[19] C. Bouras, A. Papazois and N. Stasinos, "A Framework for Cross-Platform Mobile Web Applications Using HTML5," in 2014 2nd International Conference on Future Internet of Things and Cloud (FiCloud), 2014 .

[20] Sharieh, M. Qatawneh, W. Almobaideen and A. Sleit, "Hex-Cell: modeling, topological properties and routing algorithm," European Journal of Scientific Research, vol. 22, no. 2, pp. 457-468, 2008 .

[21] M. Taguinod, A. Doup, Z. Zhao and G.-J. Ahn, "Toward a Moving Target Defense for Web Applications," in 2015 IEEE 16th International Conference on Information Reuse and Integration (IRI), San Francisco, CA, USA, 2015 .

[22] G. Wang, "Improving Data Transmission in Web Applications via the Translation between XML and JSON," in CMC, 2011, Communications and Mobile Computing, International Conference on, 2011 .

[23] K. Kumamoto, T. Amagasa and H. Kitagawa, "A System for Querying RDF Data Using LINQ," in 2015 18th International Conference on Network-Based Information Systems (NBiS), 2015 .

[24] J. R. Lourenço, B. Cabral, P. Carreiro, M. Vieira and J. Bernardino, "Choosing the right NoSQL database for the job: a quality attribute evaluation," Journal of Big Data, vol. 2, no. 18, pp. 1-26, 2015 .

[25] A. Sleit, S. Abusharkh, R. Etoom and Y. Khero, "An enhanced semi-blind DWT–SVD-based watermarking technique for digital images," The Imaging Science Journal, vol. 60, no. 1, pp. 29-38, 2012 .

[26] S. Sakr, "Cloud-hosted databases: technologies, challenges and opportunities," Cluster Computing, vol. 17, no. 2, pp. 487-502, 2014 .

[27] R. Catt, "Scalable SQL and NoSQL data stores," ACM SIGMOD Record, vol. 39, no. 4, pp. 12-27, 2010 .

[28] J. Rith, P. S. Lehmayr and K. Meyer-Wegener, "Speaking in tongues: SQL access to NoSQL systems," in SAC '14 Proceedings of the 29th Annual ACM Symposium on Applied Computing, 2014 .

[29] C.-O. Truica, A. Boicea and I. Trifan, "CRUD Operations in MongoDB," in 2013 International Conference on Advanced Computer Science and Electronics Information, 2013 .

[30] Sattar, T. Lorenzen and K. Nallamaddi, "Incorporating NoSQL into a database course," ACM Inroads, vol. 4, no. 2, pp. 50-53, 2013 .

[31] V. Guimaraes, F. Hondo, R. Almeida, H. Vera, M. Holanda, A. Araujo, M. E. Walter and S. Lifschitz, "A study of genomic data provenance in NoSQL document-oriented database systems," in 2015 IEEE International Conference on Bioinformatics and Biomedicine (BIBM), 2015 .

[32] W. Almobaideen, D. Al-Khateeb, A. Sleit, M. Qatawneh, K. Qadadeh, R. Al-Khdour and H. A. Hafeeza, "Improved Stability Based Partially Disjoint AOMDV," Int'l J. of Communications, Network and System Sciences, vol. 6, no. 5, pp. 244-250, 2013 .

[33] S. Kimak, D. J. Ellman and D. C. Laing, "An Investigation into Possible Attacks on HTML5 IndexedDB and their Prevention," in 13th Annual Post-Graduate Symposium on The Convergence of Telecommunications, Networking and Broadcasting (PGNet 2012), 2012 .

[34] Mozilla Developer Network and individual contributors, "Basic concepts - Web APIs | MDN," [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/API/IndexedDB_API/Basic_Concepts_Behind_IndexedDB. [Accessed 29 April 2016].

[35] H. Lampesberger, "Technologies for Web and cloud service interaction: a survey," Service Oriented Computing and Applications, pp. 1-40, 2015 .

[36] C. Jackson, A. Bortz, D. Boneh and J. C. Mitchell, "Protecting browser state from web privacy attacks," in WWW '06 Proceedings of the 15th international conference on World Wide Web, Edinburgh, Scotland, 2006 .

[37] R. Shaw and P. Golden, "Taking entity reconciliation offline," Proceedings of the American Society for Information Science and Technology, vol. 50, no. 1, pp. 1-4, 2014 .

[38] V. Kumar, "Local Data Access: I: IndexedDB," in Beginning Windows 8 Data Development, Apress, 2013, pp. 35-60.

[39] Mozilla Developer Network and individual contributors, "IDBObjectStore - Web APIs | MDN," Mozilla, 11 March 2016. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/API/IDBObjectStore. [Accessed 29 April 2016].

[40] Mozilla Developer Network and individual contributors, "IDBObjectStore.add() - Web APIs | MDN," Mozilla, 11 March 2016. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/API/IDBObjectStore/add. [Accessed 29 April 2016].

[41] Mozilla Developer Network and individual contributors, "IDBObjectStore.get() - Web APIs | MDN," Mozilla, 16 March 2016. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/API/IDBObjectStore/get. [Accessed 29 April 2016].

[42] Mozilla Developer Network and individual contributors, "IDBObjectStore.put() - Web APIs | MDN," Mozilla, 16 March 2016. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/API/IDBObjectStore/put. [Accessed 29 April 2016].

[43] Mozilla Developer Network and individual contributors, "IDBObjectStore.delete() - Web APIs | MDN," Mozilla, 16 March 2016. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/API/IDBObjectStore/delete. [Accessed 29 April 2016].