ISSN (online) : 2277-5390
ISSN (Print): 2277 - 5382

Object oriented JavaScript generalization technique

F. Fawzia Khan[*1], Dr.K.Thilagam[2]

[*1]*Research Scholar, Department of Computer Science, Karpagam University, Coimbatore, Tamil Nadu, India*
[2]*Associate Professor, Department of Computer Applications, Karpagam University, Coimbatore, Tamil Nadu, India*
[*1]fawziakhanphd@gmail.com

Abstract

Background/Objectives: To improve the performance of the browsers to support all functions of JavaScript by introducing an efficient generalization technique.
Methods/Statistical analysis: Object Oriented JavaScript is a high-level, dynamic and interpreted programming language used in the development of web pages and browsers as it provides user friendly environment. However there are some JavaScript function exceptions that are unrecognized by the specified browsers causing performance degradation. Hence in this paper, this problem in the browsers can be overcome by introducing a technique called Object Oriented JavaScript Generalization (OOJG) technique.
Findings: In OOJG technique first the unrecognized functions are analyzed in each browser while using different JavaScript programs. Then the generalization of functions is performed in which the related functions are used instead of the unrecognized functions to obtain the desired performance.
Application/Improvements: The experimental results also show that the performance of the browsers can be maintained with the proposed OOJG technique in an efficient manner.
Keywords: Object Oriented JavaScript Generalization, Chrome, Internet Explorer, Firefox

1. Introduction

Java script is a widely used high level dynamic programming language but highly user friendly and hence utilized for the World Wide Web content production and the web page development. JavaScript has no classes but supports constructors and prototyping to share functionality of code. Java script are highly dynamic and it is used to add multimedia contents to the web pages and can show, hide, change, resize images, and create image rollovers [1]. Java script is an Object oriented language that considers the web page elements as objects with specific properties thus providing the advantage of creating user friendly environment. JavaScript platforms provide dynamic client-side Web applications because of its combination of expressiveness, performance, and host interoperability [2]. JavaScript is an important part of the web browsers as it can only be written inside of HTML language which provides a user friendly flexible environment to the users accessing the web page for information.

The use of JavaScript in the web browsers enables user friendly interface and faster browsing performance. Advanced web browsers and mobile browsers use new techniques like Browser side template (BST) [3] for the development of web pages but still the JavaScript based HTML code has been the most famous among the web developers. There have been many high level browsers with efficient performance such as Chrome, Internet Explorer, Opera, Safari, Firefox, etc. Each of these browsers has different features and hence may provide varying performance. But the main objectives of the browsers do not change. However in some cases, the browsers may provide varying responses due to inability to recognize some JavaScript functions. This problem has been considered as the motivation for this research work.

In this paper, the problem of browser performance degradation due to the unrecognized functions is considered and a generalization technique called as Object Oriented JavaScript Generalization (OOJG) technique is introduced by using related functions. The unrecognized functions are detected in JavaScript programs and they are replaced by a related function to achieve efficient performance. The evaluation of the proposed OOJG technique is performed with the use of three browsers namely Chrome, Internet Explorer and Firefox. The results show that the OOJG technique can ensure the fair performance of the browsers.

The rest of the paper is organized as follows: Section 2 describes the related works. The proposed OOJG technique is explained in section 3. The experimental results are given in section 4 while the conclusion of the research is presented in section 5.

ISSN (online) : 2277-5390
ISSN (Print): 2277 - 5382

2. Related Works

In [4] Charles Reis et al suggested a solution for isolating the JavaScript web programs from the browser architecture in order to reduce the robustness. The authors presented abstractions of web programs and program instances to analyze the browser components interaction and appropriate program boundaries. Then the backwards compatibility tradeoffs that constrain the web content can be divided into programs without disrupting existing web sites. Finally a multi-process browser architecture that isolates the web program instances from each other is introduced for improving fault tolerance, resource management, and performance of the browsers.

In [5] Leo A. Meyerovich et al proposed technique called Conscript for specifying and enforcing fine-grained security policies for JavaScript programs in the browser. Conscript is a client-side advice implementation for security built on top of Internet Explorer 8. It allows the hosting page to express fine-grained application-specific security policies that are enforced at runtime. The approach introduces security aspects, continuous checking, police generation and automatic evaluation to improve the JavaScript browser performance with high secured policies.

In [6] Adonis P.H. Fung et al presented HTTPS Lock for the enforcement of HTTPS protection in the unmodified JavaScript browsers. The authors suggested HTTPS Lock can be deployed in websites with valid certificates by simply including several HTML and JavaScript (JS) files. When a same URL is later revisited, the cached file resided on the client-side can then enforce the use of HTTPS and forbid users to embrace any invalid certificates. When an active attacker denies the service with an invalid certificate, HTTPS Lock skips the default warning and instead displays a non-by-passable warning to the user. Thus the approach can improve the performance and security of the object oriented JavaScript browsers.

In [7] Jan Kasper Martinsen et al presented a technique called Thread-level speculation (TLS) to improve the JavaScript performance in the web applications and web browsers. Thread-level speculation (TLS) aims to dynamically extract parallelism from a sequential program to improve the overall performance of the JavaScript files. The drawback is the storage of the state of the JavaScript engine at the speculation point can increase the memory overhead.

In [8] Reginald Cushing et al also presented a technique to improve the performance of JavaScript in the web and mobile browser. The approach called Weevil Scout performs better in the web browsers and considerably in the mobile browsers to improve the security of the JavaScript than maintaining its fair performance. The major drawback is the unconvincing performance in mobile devices due to lack of power supply.

In [9] Calin Cascaval et al discussed about the need for concurrency in the mobile browsers. The JavaScript based mobile browsers do not sufficiently allow parallel browsing or multiple tabs processing in separate process resources. The major problem is the dominance of the JavaScript platforms. Most mobile browsers do not depend on CSS, rendering and parse engines for the browser performance which reduces the ability for parallelism. The authors suggested building browser architecture with integrated languages so that concurrency is achieved.

In [10] Javier Verdu et al presented Web workers API for the evaluation of the performance scalability of the JavaScript applications including web browsers. The classification of web apps according to the worker execution models is performed to analyze the effective analysis of the performance scalability.

In our previous work [11] we presented a bug analysis approach using TSVM that helps in detecting the vulnerabilities that hinder the efficient performance of object oriented JavaScript. In [12] Isatou Hydara et al proposed an approach for removing the cross-site scripting vulnerabilities from the programming languages. In [13] S. Mithun brindha et al presented a fuzzy based approach for prioritizing the security requirements in the development of software applications and web applications using the programming languages.

3. Object oriented JavaScript generalization technique

The web browsers generally use JavaScript files as they provide user friendly environment. However with different needs and different levels of user interface, there are many number of web browsers each providing its own version of comfort browsing. Thus the web users are not confined to using a default browser. This scenario causes different browsers with varying features used for the same function of web browsing. Here the problem of unrecognized JavaScript function arises. Some of the JavaScript functions may not be supported by all the browsers due to the browsers varying architecture and other characteristic features. The main idea suggested in this paper to resolve this drawback is to use a generalization technique of modifying or replacing the unsupported functions.

Based on the suggested solution, Object Oriented JavaScript Generalization (OOJG) technique is introduced in this paper to overcome the problem of unrecognized functions of the JavaScript files. In OOJG technique, the JavaScript

HTML files are analyzed and the files are run in the browsers. When executing these HTML files, if all the functions are supported the corresponding web page will be displayed by the browser. If the files contain unsupported functions the browser will not display the web page and instead displays the error page showing that the object or function is not supported by the specified browser.

In this research, the browsers namely Chrome, Internet Explorer and Firefox are utilized for the analysis of the proposed OOJG technique. The features and functionalities of the three browsers are briefly studied [14] and then the analysis of the unrecognized functions is performed.

3.1 Google Chrome

Google Chrome is developed by Google as a freeware web browser using Webkit layout engine and Webkit fork Blink. It was initially released as a beta version of for Microsoft Windows. Chrome was assembled by using 25 different code libraries from Google and also from third parties such as Mozilla's Netscape Portable Runtime, Network Security Services, NPAPI Skia Graphics Engine, SQLite, and many other open-source projects. The JavaScript virtual machine is one of the important projects which are mostly utilized for the web applications such as Gmail for faster processing. Chrome uses Blink rendering machine for displaying the web pages by using its own multi-process architecture and also the web core components of the Webkit. The chrome coding has to pass the web kit layout tests, fuzz tests and the automated user interface testing of scripted user actions.

Chrome has distinct features of minimalistic user interface and strong browser performance. Chrome allows synchronization of bookmarks, history and history with an additional feature of signed-in updation. The web standards such as JavaScript/ECMAscript are well supported by Chrome and also the security and privacy browsing features of Chrome provide comfortable usage of the browser. The most important features of Chrome are the stable and faster user interface architecture that provides highly upgraded applications to be performed effectively in Chrome. Chrome is available for Windows, OS X, Linux, Android and iOS platforms.

Using OOJG technique in Chrome helps in detecting the unsupported JavaScript functions and then the generalization of functions is performed. The unsupported

3.2. Internet Explorer

Internet Explorer (IE) was developed by Microsoft Windows as a series of graphical web browsers initially included as a part of Microsoft Windows OS. IE used source code from Spyglass, Inc. Internet Explorer has been designed to view a broad range of web pages and provide certain features within the operating system, including Microsoft Update. Internet Explorer uses a componentized architecture built on the Component Object Model (COM) technology. It consists of several major components, each of which is contained in a separate Dynamic-link library (DLL) and exposes a set of COM programming interfaces hosted by the Internet Explorer main executable, iexplore.exe. Pop-up blocking and tabbed browsing were the recent features added along with favicon feature. IE supports HTML 4.01, HTML 5, CSS Level 1, Level 2 and Level 3, XML 1.0, and DOM Level 1, with minor implementation gaps and completely supports XSLT 1.0 and WD-xsl. IE also uses a zone-based security framework that websites based on certain conditions for user-editable and secured zones. Security restrictions are applied per zone and all the sites in a zone are subject to the restrictions.

3.3. Firefox

Firefox has been developed by Mozilla foundation supported for Windows, Linux, OS X, Android and Firefox OS. Features include tabbed browsing, spell checking, incremental find, live bookmarking, Smart Bookmarks, a download manager, private browsing, and location-aware browsing based on a Google service. Firefox is an integrated search system that uses Yahoo Search by default in most localization. Firefox also provides an environment for web developers in which they can use built-in tools, such as the Error Console or the DOM Inspector, or extensions, such as Firebug and more recently there has been an integration feature with Pocket. Firefox Hello was an implementation allowing 2 users of Firefox to have a video call, with the extra feature of screen/file sharing, by sending a link to each other.

Functions can be added through add-ons created by third-party developers. Add-ons are primarily implemented by means of the XUL and XPCOM APIs, which allow them to directly access and manipulate much of the browser's internal functionality. The new versions of Firefox for mobiles also includes the Awesome bar, Add-on support, password manager and the ability to synchronize with the user's computer Firefox browser using Firefox Sync.

ISSN (online) : 2277-5390
ISSN (Print): 2277 - 5382

3.4 Algorithm: OOJG Technique

Input: JavaScript files with specified functions
Output: Unsupported JavaScript functions and Fair Browser performance
Step 1: Initialize input files with functions
//Event function, indexof function, onfocus function, tosource function, trim functions
Browsers = Chrome, Internet Explorer, Firefox
Step 2: Run each file in each Browser
If (Successful execution)
File has no unsupported functions
Else if
Object is not supported
//File has unsupported functions
End if
Step 3: Initialize OOJG technique
Find unsupported function ()
Use alternate function ()
Step 4: Re-run the file with unsupported function ()
If (Successful execution)
File has no unsupported functions
Else if
Go to step 3
End if
End

3.5. Description: The JavaScript files are initialized and the set of browsers to be utilized are considered. Each file is run individually on each browser. If the file is executed successfully, then the file is completely supported. But if the execution is not successful then there are some unsupported functions in the executed file. At this case, the OOJG technique is initialized to find the unsupported functions and then they are replaced by alternate coding functions. Then the modified coding is again run on individual browsers to determine the effectiveness of the new functions. Thus the OOJG can provide fair performance of the JavaScript browsers.

The implementation of the proposed OOJG technique initially analyzes the browsers for the JavaScript HTML support and then analyzes the JavaScript files. Then the files are run on the browsers and the functions that are not supported by the browsers are identified. Then the possible solutions for executing the unsupported functions are carried out. In this research the replacement of unsupported functions by a related function is used as the possible strategy for executing the unsupported functions. This strategy results in the successful execution of the files and leads to the corresponding web pages. Thus the proposed OOJG technique can provide efficient execution of the files with unsupported functions to achieve better performance.

4. Experimental Results

The experiments are performed by executing some of the JavaScript HTML files in the selected browsers namely Chrome, Internet Explorer and Firefox. From the results, the unsupported functions of the JavaScript files are identified. Then the related functions are implemented to replace the unrecognized functions inorder to execute the JavaScript files.

In this research, inorder to evaluate the performance of the proposed OOJG technique, five JavaScript files containing functions namely event function, indexof function, onfocus function, tosource function and trim functions are considered. These functions are executed in the three browsers to analyze which of the functions are not supported in each browser. Then based on the analysis results the suitable functions are used instead of the unsupported functions.

4.1. Event function

The file containing the event function is executed on the three browsers. The addEventListener () function is executed successfully in the chrome and Firefox but it is found not supportable in IE.

ISSN (online) : 2277-5390
ISSN (Print): 2277 - 5382

Figure.1. Unsupported coding of event function

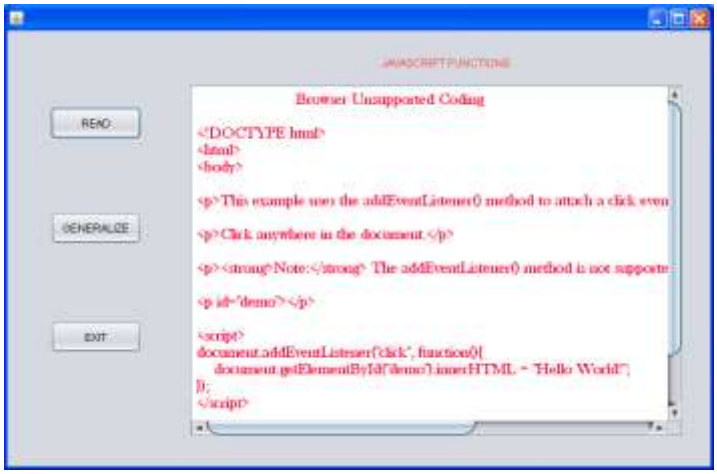
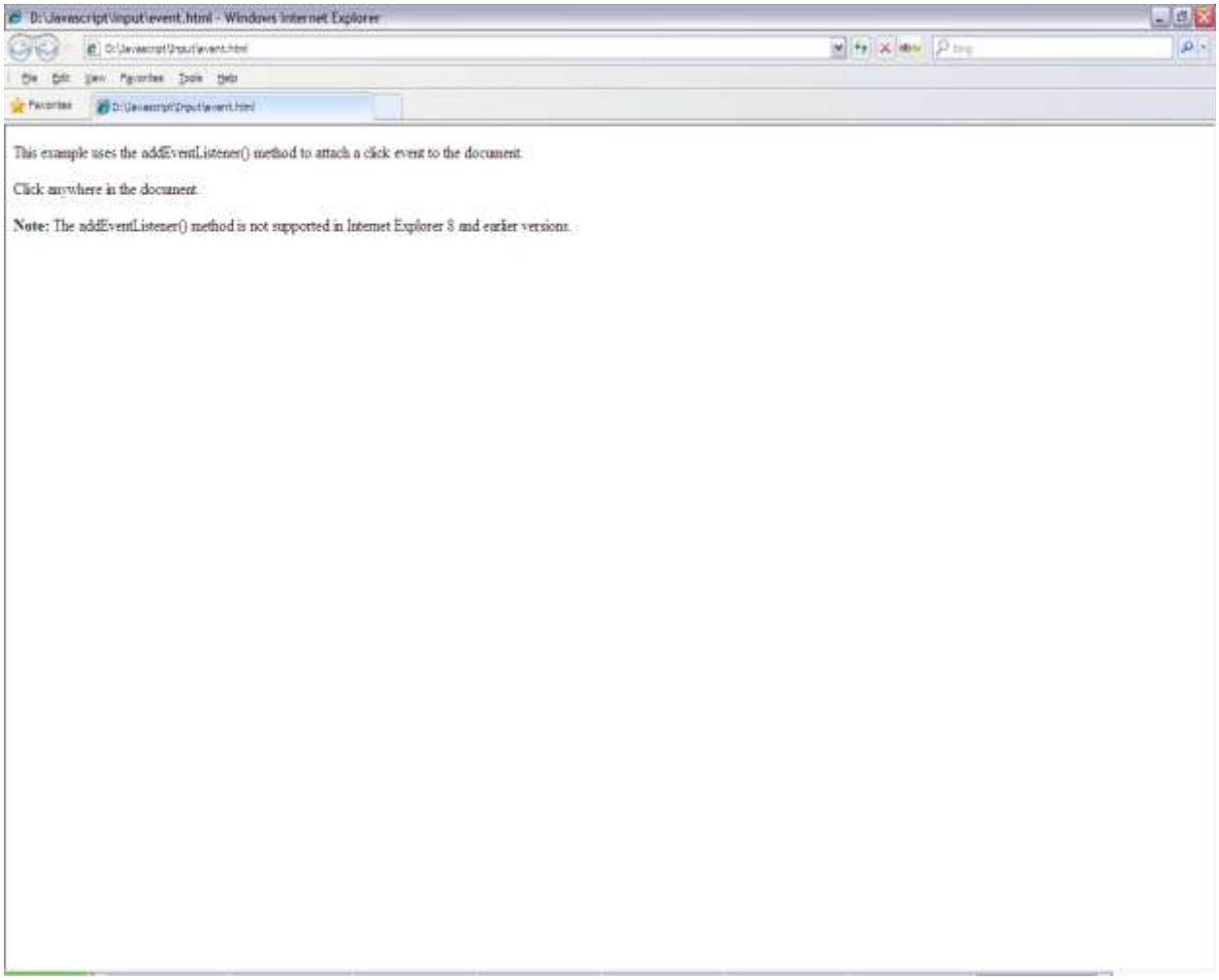


Figure 1 shows the unsupported coding of the JavaScript HTML file containing the event function in the IE browser. From the figure, it is clear that the event function (addEventListener ()) is not supported in the IE.

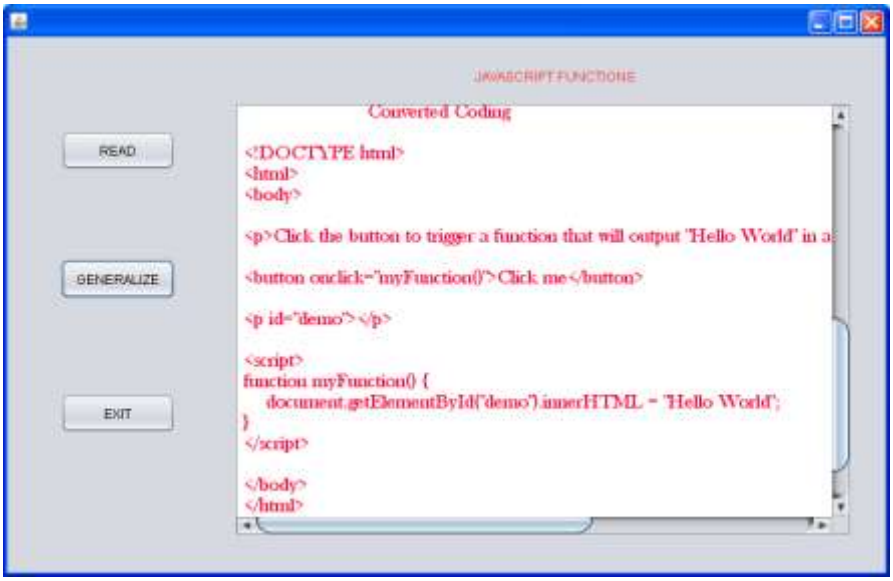
Figure.2. Output of event function file



Hence the OOJG technique uses the replacement strategy by using a function called onclick to execute the file in place of event function. Figure 2 shows the output of the event function. Then by using the onclick function, the desired performance of event function file that was achieved in Chrome and Firefox can be achieved. The modified coding is shown in Figure 3.

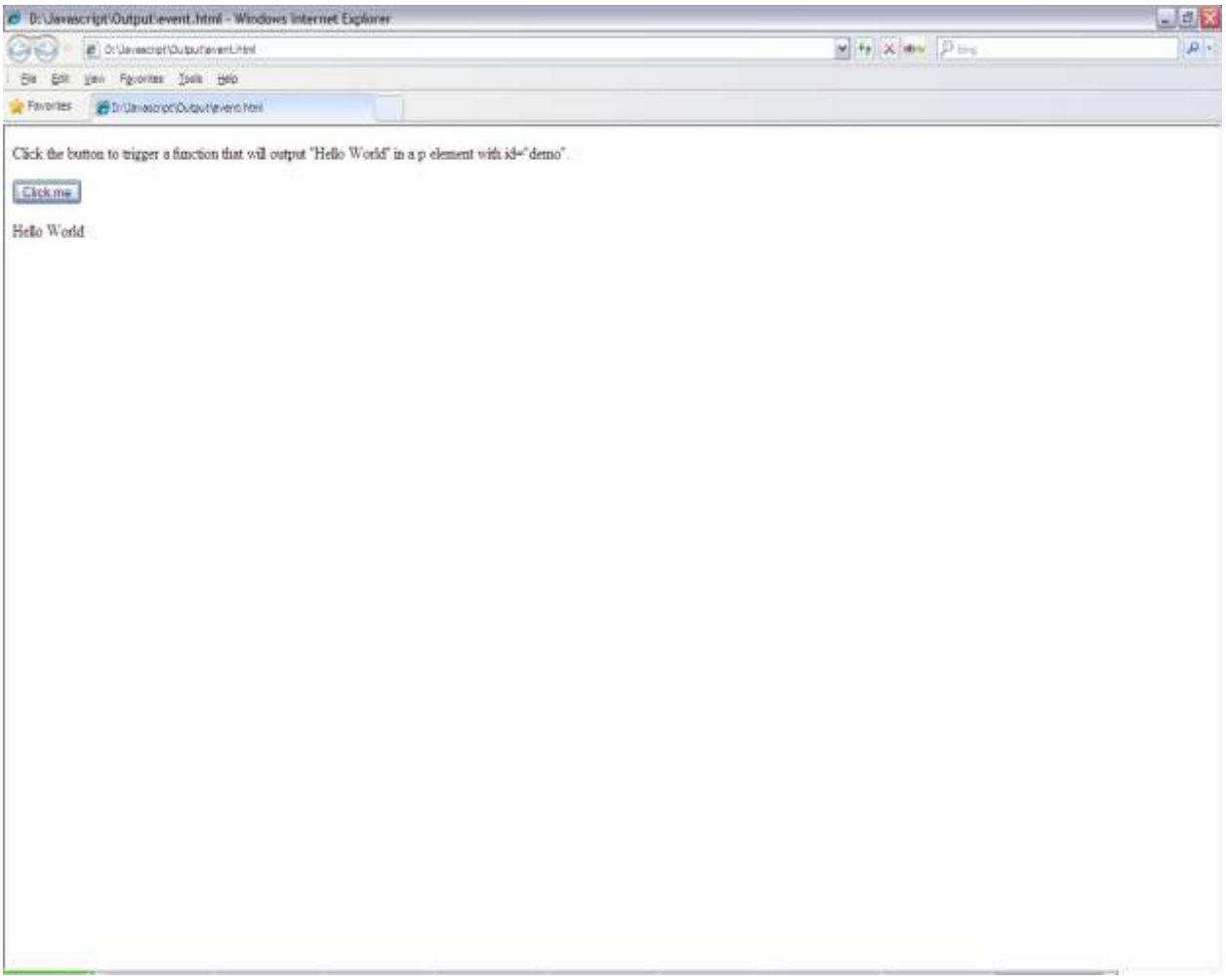
ISSN (online) : 2277-5390
ISSN (Print): 2277 - 5382

Figure.3. Modified coding with replaced function



The final output of the event function JavaScript file with the replaced onclick function is shown in figure 4.

Figure.4. Final Output



4.2. Indexof function

The file containing the indexof function is executed on the three browsers. The indexof () function is executed successfully in the chrome and Firefox but it is found not supportable in IE. So the function has to be replaced.

ISSN (online) : 2277-5390
ISSN (Print): 2277 - 5382

Figure.5. Unsupported coding of indexof function

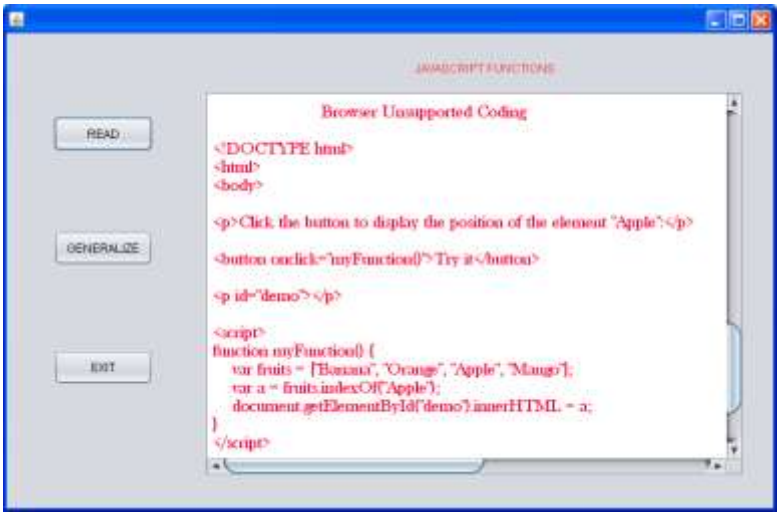
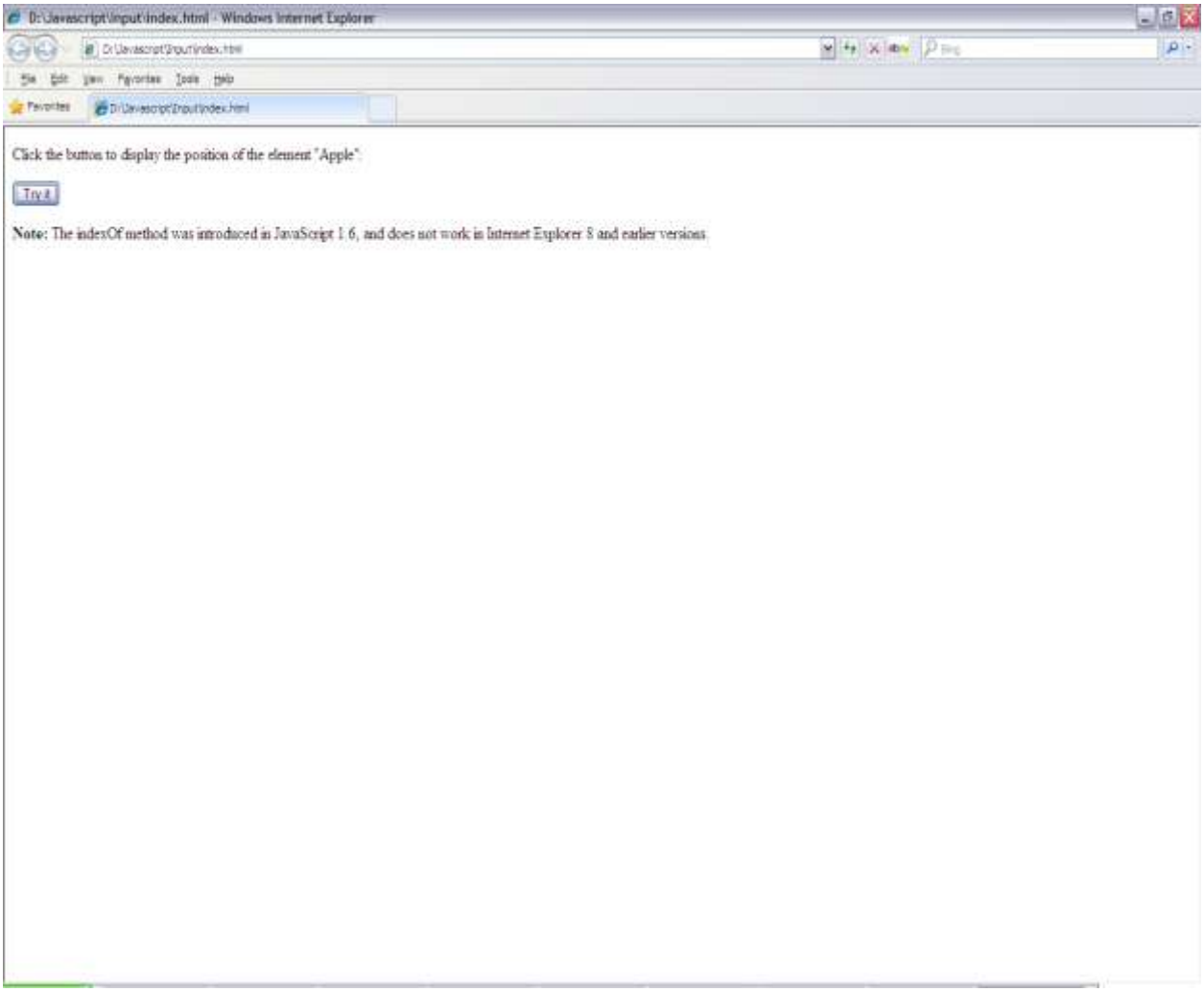


Figure 5 shows the unsupported coding of the JavaScript HTML file containing the indexof function in the IE browser. From the figure, it is clear that the indexof function is not supported in the IE.

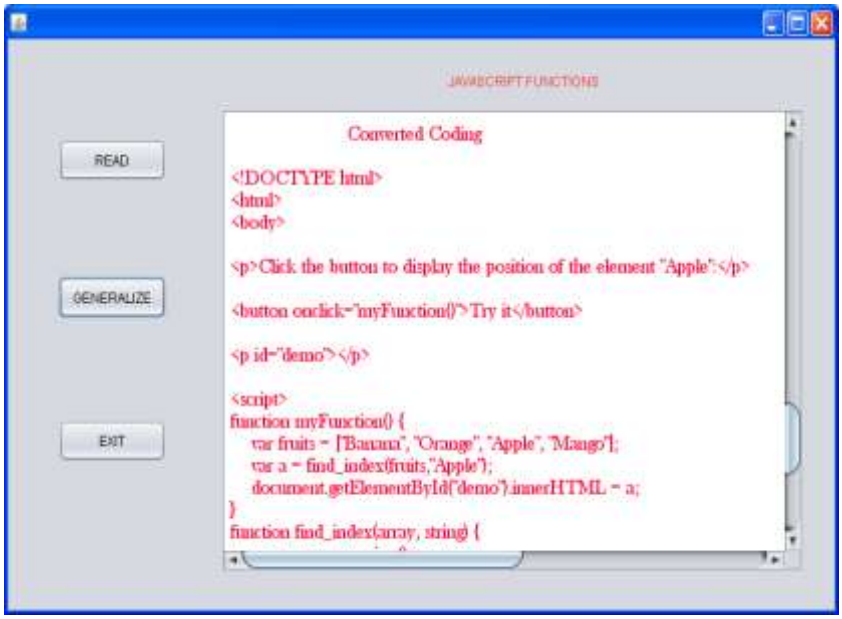
Figure.6. Output of indexof function



Hence the OOJG technique uses the replacement strategy by using separate function coding to execute the file. Figure 6 shows the output of the indexof function. Then the desired performance of indexof function file that was achieved in Chrome and Firefox can be achieved. The modified coding with the replaced coding is shown in figure 7.

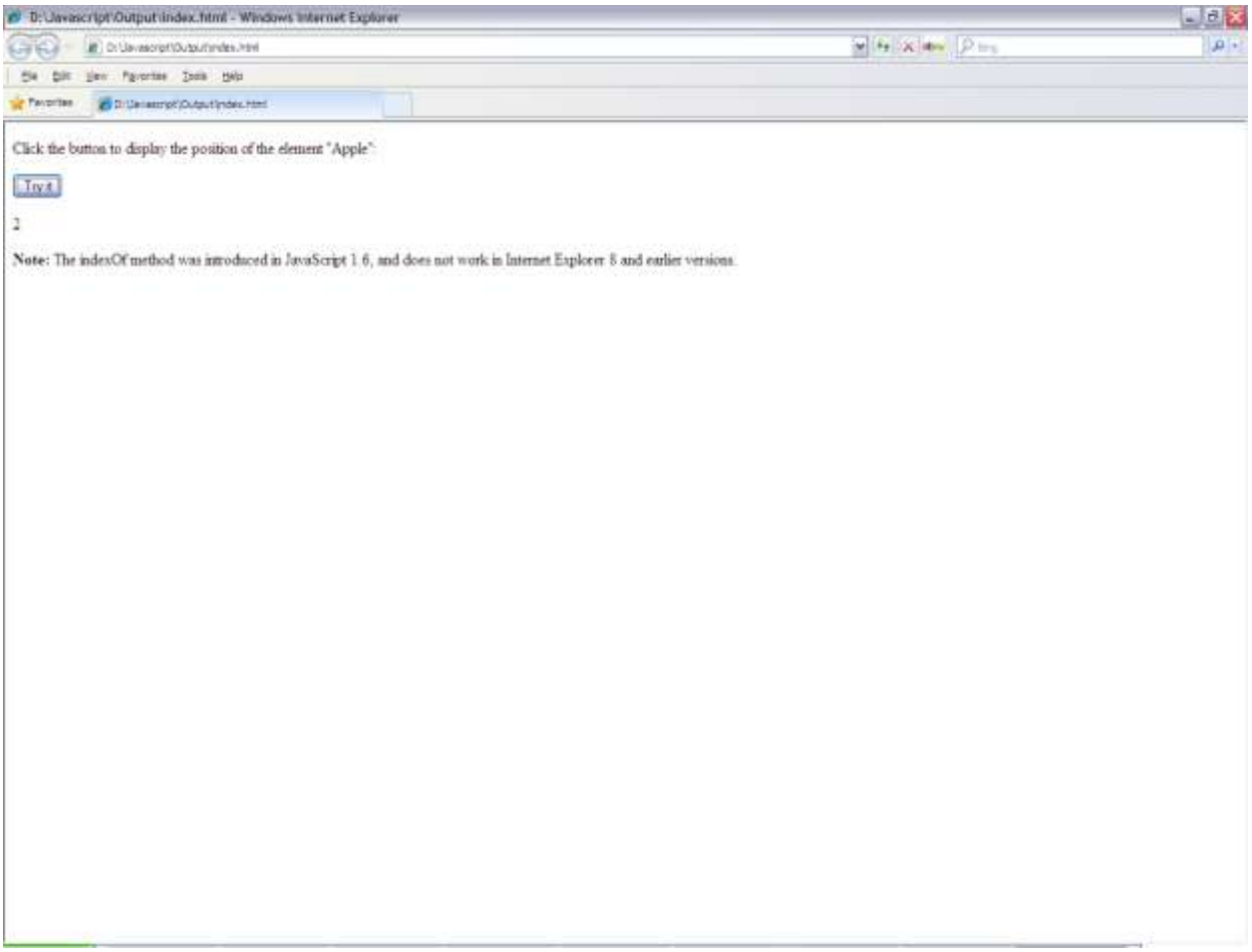
ISSN (online) : 2277-5390
ISSN (Print): 2277 - 5382

Figure.7. Modified coding with replaced function coding



The final output of the indexof function JavaScript file with the replaced function coding is shown in figure 8.

Figure.8. Final output of Modified coding



4.3 Onfocus function

The file containing the onfocus function is executed on the three browsers. The onfocus function is executed successfully in the chrome and IE but it is found not supportable in Firefox. So the function has to be replaced.

ISSN (online) : 2277-5390
ISSN (Print): 2277 - 5382

Figure.9. Unsupported coding of onfocus function

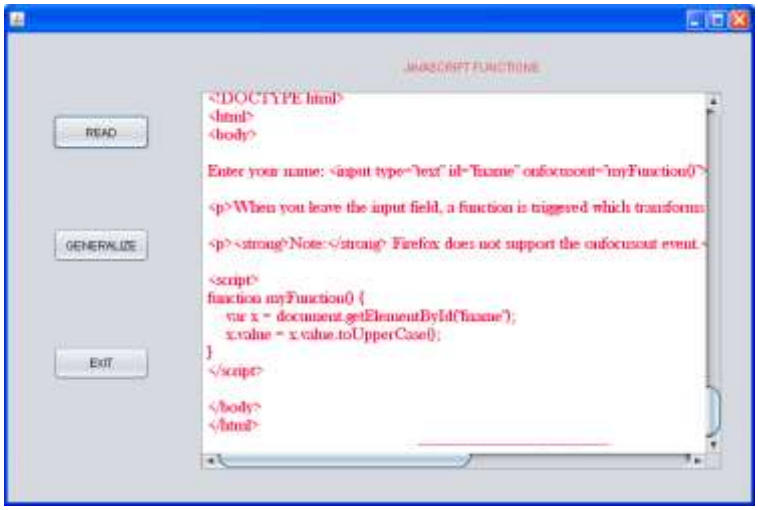


Figure 9 shows the unsupported coding of the JavaScript HTML file containing the onfocus function in the Firefox browser. From the figure, it is clear that the onfocus function is not supported in the Firefox.

Figure.10. Output of onfocus function

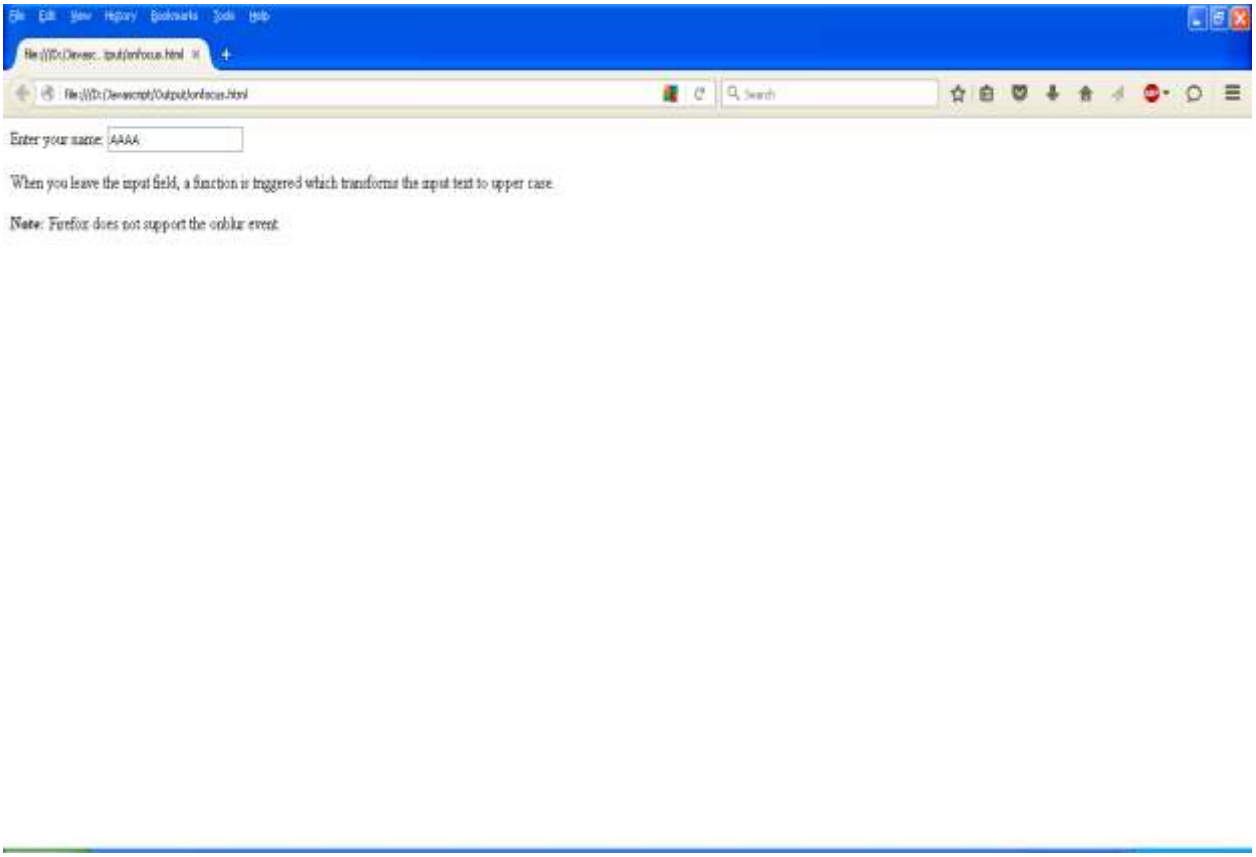
Hence the OOJG technique uses the replacement strategy by using onblur function to execute the JavaScript file. Figure 10 shows the output of the onfocus function. Then the desired performance of onfocus function file that was achieved in Chrome and IE can be achieved in Firefox. The modified coding with onblur function is shown in figure 11.

Figure.11. Modified coding with replaced onblur function

ISSN (online) : 2277-5390
ISSN (Print): 2277 - 5382

The final output of the onfocus function JavaScript file with the replaced onblur function is shown in figure 12.

Figure.12. Final output with modified coding



4.4 Tosource function

The file containing the tosource function is executed on the three browsers. The tosource function is executed successfully in the Firefox and IE but it is found not supportable in Google Chrome. So the function has to be replaced by a related coding.

ISSN (online) : 2277-5390
ISSN (Print): 2277 - 5382

Figure 13. Coding of tosource function

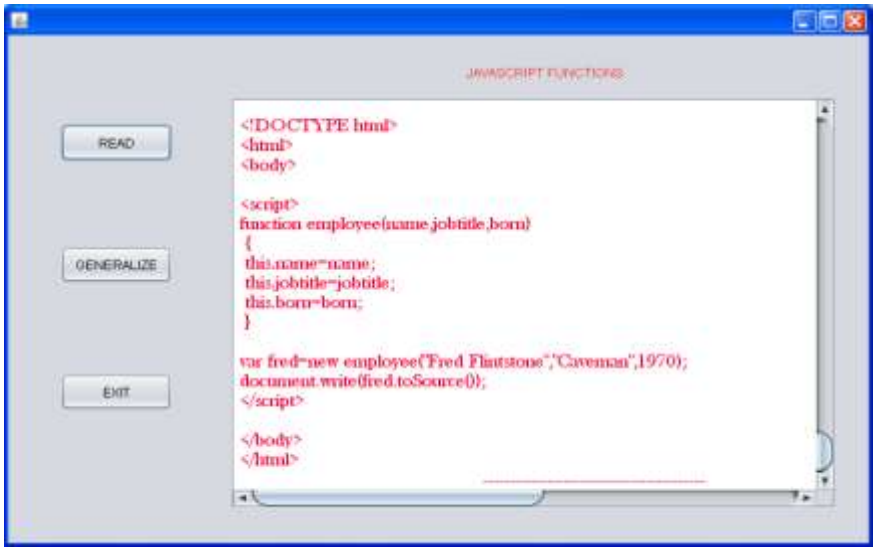
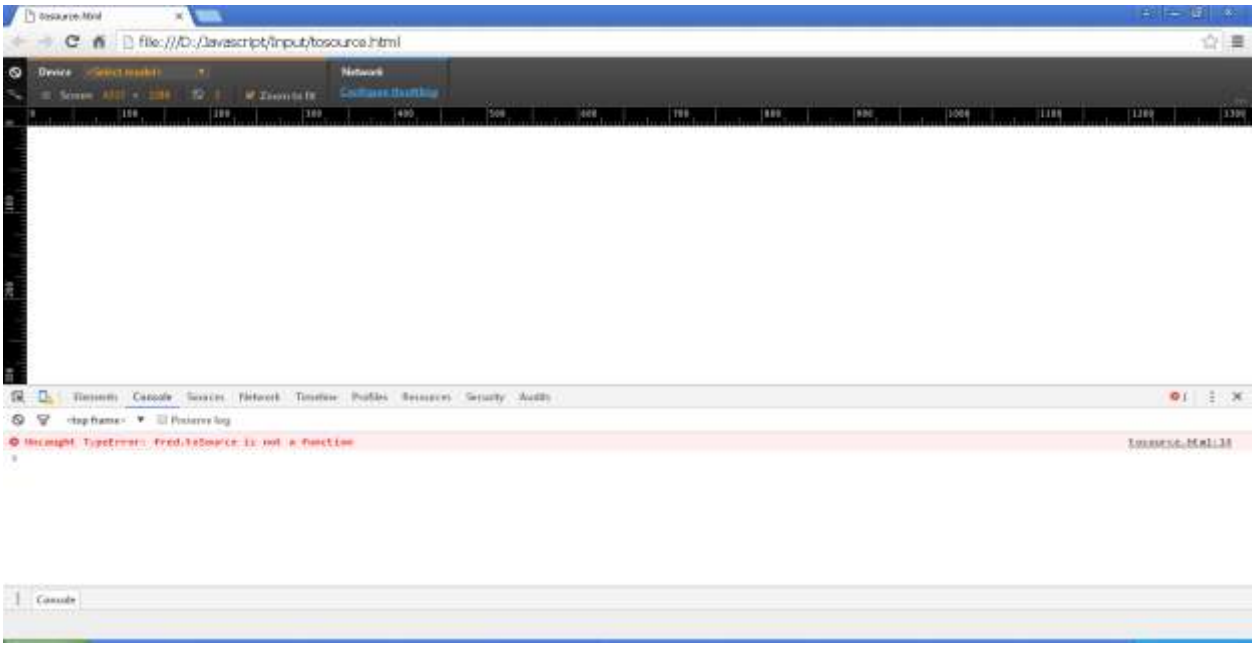


Figure 13 shows the unsupported coding of the JavaScript HTML file containing the tosource function in the Chrome browser. From the figure, it is clear that the tosource function is not supported in the Chrome.

Figure.14. Output of tosource function



Hence the OOJG technique uses the replacement strategy by using JSON.stringify function to execute the JavaScript file. Figure 14 shows the output of the tosource function.

Then the desired performance of tosource function file that was achieved in Firefox and IE can be achieved in Chrome. Figure 15 shows the modified coding with replacement for tosource.

The final output of the tosource function JavaScript file in Chrome with the replaced JSON.stringify function is shown in figure 16.

ISSN (online) : 2277-5390
ISSN (Print): 2277 - 5382

Figure.15. Modified coding with replaced JSON.stringify function

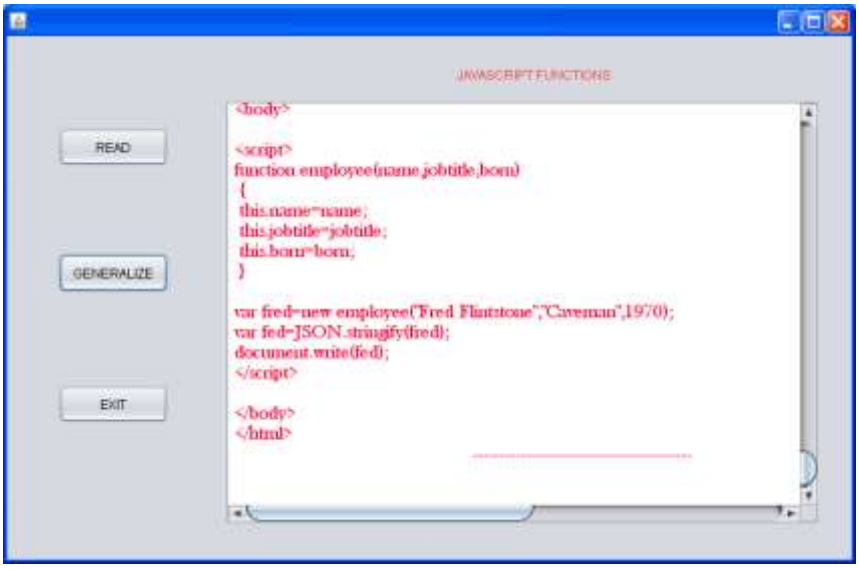
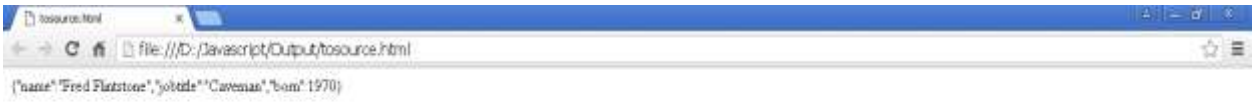


Figure.16. Final output with Modified coding



4.5 Trim function

The file containing the trim function is executed on the three browsers. The trim function is executed successfully in the Chrome and Firefox but it is found not supportable in IE. So the function has to be replaced by a related coding.

Figure.17. Unsupported coding of trim function

ISSN (online) : 2277-5390
ISSN (Print): 2277 - 5382

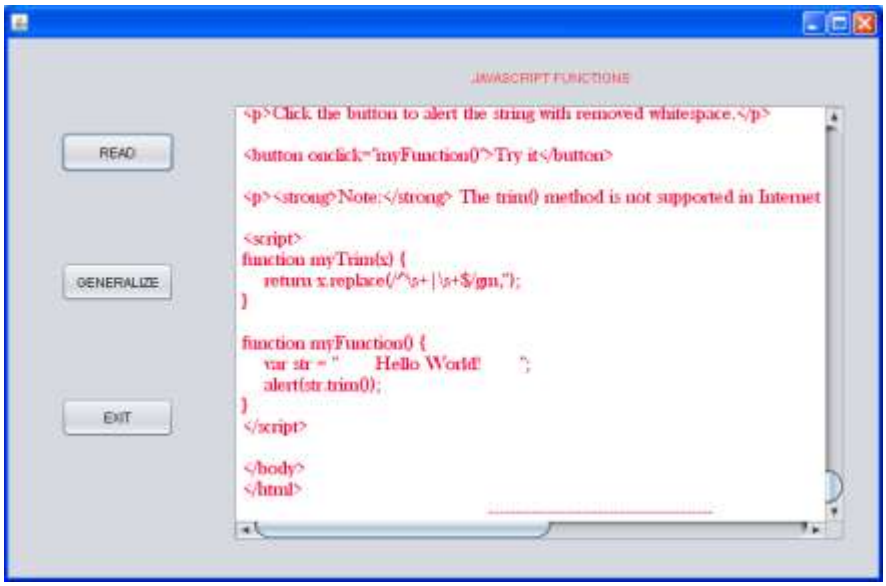
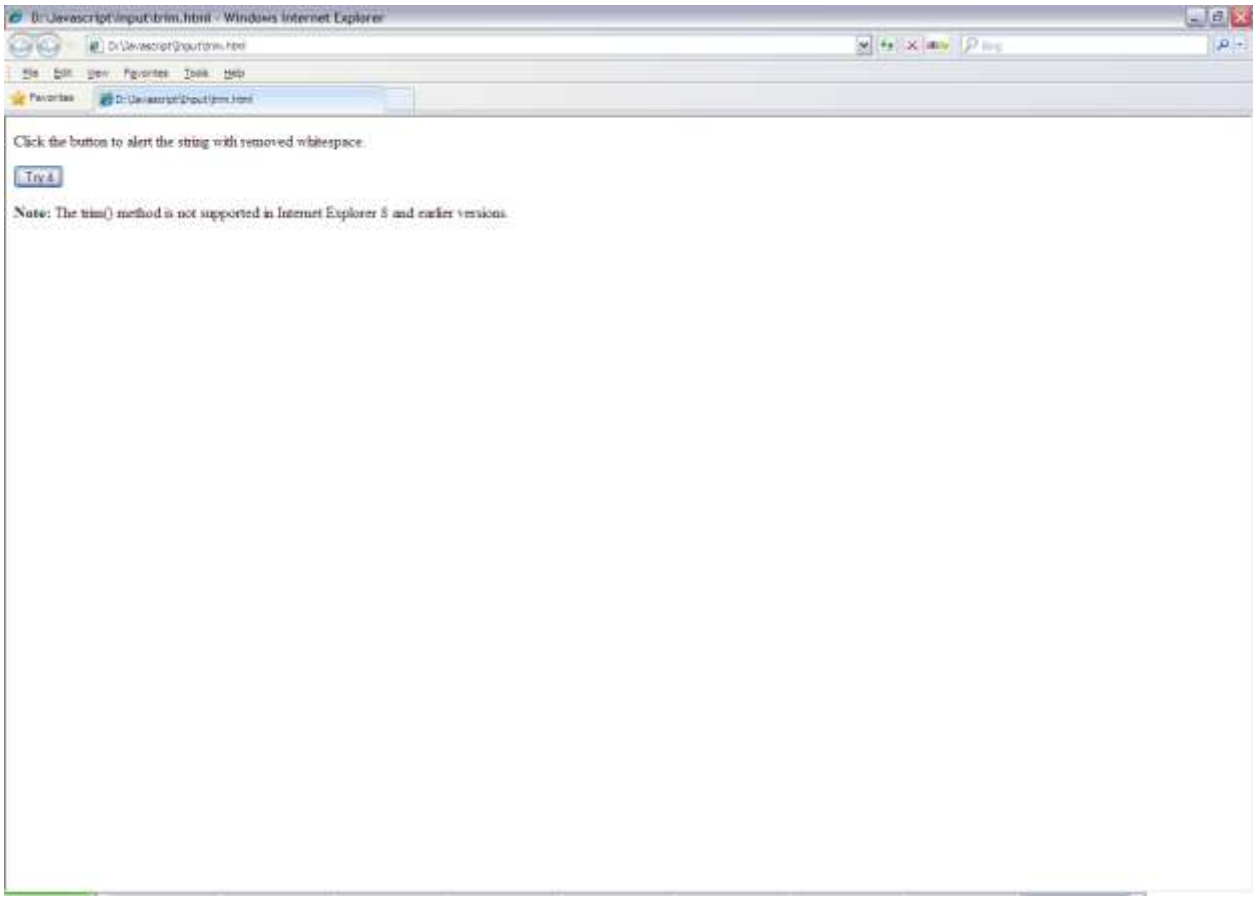


Figure 17 shows the unsupported coding of the JavaScript HTML file containing the trim function in the IE browser. From the figure, it is clear that the trim function is not supported in the IE.

Figure18. Output of trim function

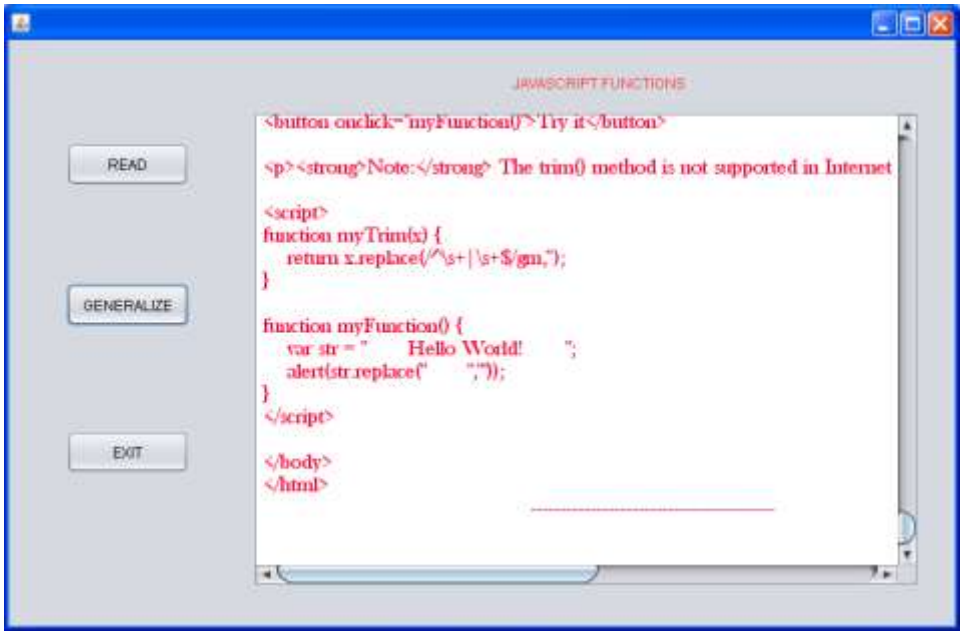


Hence the OOJG technique uses the replacement strategy by using replace function to execute the JavaScript file. Figure 18 shows the output of the trim function.

The desired performance of trim function file that was achieved in Chrome and Firefox can be achieved in IE by modifying the coding. Figure 19 shows the modified coding with the replace function.

ISSN (online) : 2277-5390
ISSN (Print): 2277 - 5382

Figure.19. Modified coding with Replace function



The final output of the trim function JavaScript file in Chrome with the modified coding is shown in figure 20.

Figure.20. Final output of Modified coding

4.6. Performance Comparison

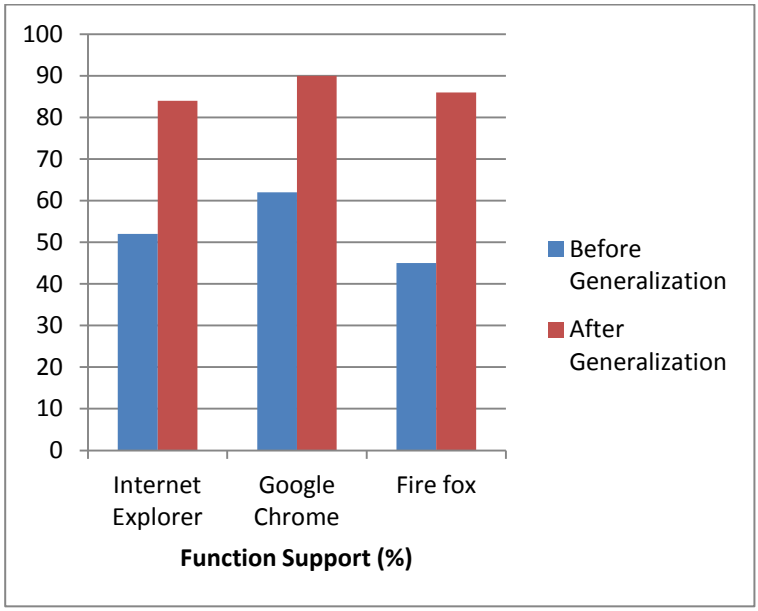
The result for the proposed Generalization technique is initialized by considering the number of functions in the Java script used in the browser. Let us consider the total number of functions be 50. The result comparison is concerned with the percentage of the java script function supported before generalization is applied and after generalization technique is applied. Consider the browsers Internet Explorer, Google Chrome and Firefox. Each browser is considered for the improvement in the number of function support after Generalization.

The percentage of function support is calculated as

$$\% \text{ Function Support } = \frac{\text{No. of functions supported}}{\text{Total No of functions used}} \times 100$$

ISSN (online) : 2277-5390
ISSN (Print): 2277 - 5382

Figure 21. Function Support Comparison



The total number of functions considered is 50, in Internet Explorer the number of functions supported before generalization is 26, and after generalization is 42. Thus the percentage of function support before generalization is 52 % and after generalization the percentage is improved to 84%. From the figure 21, it is clear that the generalization technique has improved the percentage to a greater extent. Thus the proposed generalization technique makes a unrecognized function in the object oriented java script to be supported in the browser execution and increase its performance efficiency.

5. Conclusion

Object oriented JavaScript Generalization technique has been introduced to maintain the fair performance of the browsers for the common JavaScript files irrespective of the varying features of the browsers. The proposed OOJG technique uses the replacement strategy of modifying the existing JavaScript coding by using related functions instead of the unsupported functions. This approach of OOJG achieves fair performance of the browsers even when not supporting certain functions. Thus the problem of unrecognized or unsupported functions in the web browsers can be effectively resolved with the OOJG technique.

6. References

[1] Gregor Richards, Sylvain Lebresne, Brian Burg, Jan Vitek. An analysis of the dynamic behavior of JavaScript programs. In ACM Sigplan Notices. 2010; 45(6), 1-12.
[2] Mark McGranaghan. Clojurescript: Functional programming for javascript platforms. IEEE Internet Computing. 2011; 6, 97-102.
[3] Francisco J. Garcia-Izquierdo, Raul Izquierdo. Is the browser the side for templating?. *Internet Computing, IEEE.* 2012; 16(1), 61-68.
[4] Charles Reis, Steven D. Gribble. Isolating web programs in modern browser architectures. In Proceedings of the 4th ACM European conference on Computer systems, 2009; 219-232.
[5] Leo Meyerovich, Benjamin Livshits. ConScript: Specifying and enforcing fine-grained security policies for Javascript in the browser. In 2010 IEEE Symposium on Security and Privacy (SP). 2010; 481-496.
[6] Adonis PH Fung, K. W. Cheung. HTTPS Lock: Enforcing HTTPS in unmodified browsers with cached JavaScript. In 2010 4th International Conference on Network and System Security (NSS). 2010; 269-274.
[7] Jan Kasper Martinsen, Hakan Grahn, Anders Isberg. Using speculation to enhance javascript performance in web applications. Internet Computing, *IEEE*. 2013; 17(2), 10-19.
[8] Reginald Cushing, Ganeshwara Herawan Hananda Putra, Spiros Koulouzis, Adam Belloum, Marian Bubak, Cees De Laat. Distributed computing on an ensemble of browsers. Internet Computing, *IEEE*. 2013; 17(5), 54-61.

ISSN (online) : 2277-5390
ISSN (Print): 2277 - 5382

[9] Calin Cascaval, Pablo Montesinos Ortego, Behnam Robatmili, Dario Suarez Gracia. Concurrency in mobile browser engines. Pervasive Computing, *IEEE*. 2015; 14(3), 14-19.
[10] Javier Verdu, Alex Pajuelo. Performance scalability analysis of javascript applications with web workers. *IEEE* Computer Architecture Letters, 2015; PP(99), 1.
[11] F. Fawzia Khan, R. Mallika. Analysis of various types of bugs in the object oriented java script language coding. *Indian Journal of Science and Technology*. 2015; 8(21), 1-9.
[12] Isatou Hydara, Abu Bakar Md Sultan, Hazura Zulzalil, Novia Admodisastro. Removing cross-site scripting vulnerabilities from web applications using the OWASP ESAPI Security Guidelines. *Indian Journal of Science and Technology*. 2015; 8(30), 1-5.
[13] S. Mithun brindha, Dr.G. Singaravel. Fuzzy interference approach based prioritization of security requirements. *Indian Journal of Innovations and Developments*. 2014; 3(4), 80-84.
[14] Wikipedia-https://en.wikipedia.org/

The Publication fee is defrayed by Indian Society for Education and Environment (iSee). www.iseeadyar.org

***Citation**:*

F. Fawzia Khan, K.Thilagam. Object oriented javascript generalization technique. *Indian Journal of Innovations and Developments.* 2016; 5 (4), April.