International Journal of Advanced Science and Technology
Vol.117 (2018), pp.1-10
http://dx.doi.org/10.14257/ijast.2018.117.01

The JavaScript and Web Assembly Function Analysis to Improve Performance of Web Application

Jin-Tae Park[1], Hyun-Gook Kim[2*] and Il-Young Moon[3]

[1,2,3]*Department of CSE, KOREATECH University, Cheon-an, Republic of Korea*
[1]*wlsxo05@koreatech.ac.kr*, [2*]*hy1392@koreatech.ac.kr*, [3]*iymoon@koreatech.ac.kr*

Abstract

All equipment was connected to the Internet to the 4th industrial revolution, and the application of Web technology became possible. Since it is possible to install a Web browser on the device, the importance was placed on the function of controlling the hardware via the browser. However, existing Web technologies have lower performance than native languages, and to complement this, a new type of Web code called Web Assembly was done. The web assembly plays a role of converting native code so that it can be executed by a JavaScript based engine. As a result, Web technology can be utilized in even more diverse fields. Therefore, in this paper, we will examine the effective area of two technologies in application development through performance comparison analysis when code used for developing web application is operated with web assembly and JavaScript.

Keywords: *JavaScript, Web Assembly, Web Application, Optimization Performance*

1. Introduction

Past web technology had its purpose in providing unilateral information via browser. The Web technology of the age of Web 1.0 provided basic directory search only. All data were systematically categorized and the user searched the data according to that category. Since then, the Web technology of the Web 2.0 era has come to have a form in which users directly generate information and communicate bidirectional based on the spirit of openness, participation and sharing. In other words, the environment where users can directly produce contents and communicate bi-directionally began to be constructed. Web 3.0 means an intelligent web technology that can use semantic web technology to provide personalized information by understanding contents included in web pages. In other words, intelligent, personalized custom Web era began. Web technology can integrate with the Internet's Internet and artificial intelligence, which is the core technology of the 4th industrial revolution, IoT that gives cognitive and communication skills to everything is sometimes called WoT (Web of Things) [1].

The current Web technology is preparing the Web 4.0 era beyond the Web 3.0 era. In the Web 4.0 era, humans are always started as being always on-line. In the virtual world where you are connected online at any time, that is, start a conversation with the web, the role of the browser has become important. The Web browser has no role of merely displaying information, and is a tool that can communicate with the virtual space called Web. Many of these smart devices were equipped with a new operating system called Web OS and provided an environment where applications can be provided via a browser. However, HTML, CSS, and JavaScript, which are languages for creating information and applications provided through existing Web browsers, had problems in fully representing the virtual world that the Web can provide. The JIT (Just-In-Time) compiler appeared in JavaScript, the dramatic development of the performance, the base of the Web application

Received (April 16, 2018), Review Result (June 2, 2018), Accepted (June 8, 2018)

ISSN: 2005-4238 IJAST

can become bigger and bigger. However, the problem of performance is still being presented in reality. These problems become a major obstacle that web technology can develop, and the concept of web assembly is studied to complement this problem. The web assembly means that you develop an application using an existing native language (C type) and convert it to be operable on a browser equipped with a JavaScript based engine. It is w3c and it consists of WG (Working-Group) in November 2017, and it is still under active research at present. The reason why web assemblies have better performance than JavaScript is that the code is concise, the parsing time is small, and it takes less time to compile and optimize. However, in some cases the Web Assembly may not be faster than JavaScript [2-4].

Therefore, in this paper, we will investigate the details of the Web assembly and would like to deal with the performance as to what extent it looks when compared with the existing JavaScript. Therefore, we want to build each test environment and compare function-centered comparisons.

2. Related Research

2.1. JavaScript

JavaScript is a scripting language in the programming language. Today almost all Web browsers have an interpreter built in and interpreted and executed by the JavaScript browser interpreter. HTML is responsible for the basic structure of the Web application, when the CSS is in charge of the design, the JavaScript is in charge of the application running on the client side. JavaScript is multi-paradigm language, instruction type, functional type, object oriented language. Basically it follows the functional programming paradigm. Several static web applications can be done without using Java scripts, but for complicated animation effects, JavaScript is required [5].

JavaScript is sometimes called ECMAScript after starting standardization work from 1996. Early JavaScript were operated in Web browsers and were mainly used for giving effects to Web pages and improving functions. With the appearance of Node.js, the role of Java scripts began to change. Node.js is a JavaScript framework that runs on the server. By doing so, it became possible to configure both DB - server - client with JavaScript, and made a chance to expand the area of JavaScript. And a MEAN stack (MongoDB, Express.js, Angular.js, and Node.js) that can replace the LAMP stack (Linux, MySQL, PHP / Python / Perl) was highlighted. While it is possible to unify the entire application development language with JavaScript, there is no need for special processing to transfer the DB data to the client via JSON.

In addition, researches for optimizing JavaScript are actively carried out. Typical of them there is research to shorten the load time of web application. In a web application, page loading is related to various things more than transferring just bytes. It has to run Parse, Interpret, run after the browser downloads the script. In this part, the performance of the web application will be different.

The JavaScript engine consumes considerable time to run Parsing, Compiling and scripts at application startup. The time it takes for the user to interact with the application is delayed. In addition, it was confirmed that the step of Parse, Compile causes a bottleneck of a large-scale application or framework. For mobile, parse, Compile time takes about 2 to 5 times more than desktop environment [6].

For these reasons, we could check the size of the script. There is a research result that it takes an average of 4 seconds of desktop when running an application on startup (parse / compile / exec), and parse time on mobile is more than 36% more than desktop. HTTP Archive announced the size of script for each average page is 420 kb, and the above result appeared on the premise that it complies with this size.

However, the size of the script and parse, compile time are not directly proportional. This is because the time taken for parse and compile can be displayed differently even for script files of the same size irrespective of browser, device and network.

Currently available modules of JavaScript are overflowing and optimized modules are being developed continuously. However, poorly used or selected libraries or modules can result in compile or function calls taking time on the main thread. Therefore, designing ability to select and utilize the modules necessary for the application to be created is regarded as important.

2.2. Web Assembly

Web Assembly is a new type of code that can be rotated in modern web browsers. It offers new functions and has great merit in terms of performance. It is designed to be an effective compilation target for low-level source languages such as C, C ++, Rust. Web assemblies have significant implications on the Web platform. It provides a way to execute code written in multiple languages on the Web at a speed close to native, making it possible to run client applications that could not be executed on the Web on the Web until now It is because it is done. The goals that web assembly pursues are as follows [7].

- High speed, effective, portability seems good
- Easy to read and debug able
- To maintain safety
- Do not break the web

The Web platform is a set of Web APIs(DOM, CSSOM, WebGL, Indexed DB, Web Audio API) that make it possible to execute specific tasks by calling functions of Web browser and hardware, such as Java scripts, virtual machines(VM) capable of running code making up the application Can be classified. Until now, the VM of the Web browser could only load JavaScript. However, there were performance problems in the case of various fields requiring 3D games, virtual / augmented reality, image processing, image / video editing, and other native performances [8].

Also, the cost required to download, analyze and compile large-scale JavaScript applications was used much. Performance bottlenecks also appeared markedly in environments where mobile phones and resources are limited.

Web Assembly, like assemblies, offer a near-native performance as a low-level language that has a compact binary format. Also, Web Assembly will be able to work on the Web, programs written in that language as compilation targets for languages that have low level memory model like C ++ or Rust. It was developed to use the advantages of both web assembly and JavaScript like. The main concept of the web assembly is as follows, and all concepts are reflected in the JavaScript API of the Web Assembly [9-10].

- Module: Represents a web assembly binary compiled into machine code executable in the browser. Modules can be cached in Indexed DB explicitly like Blobs because there is no state and can be shared among windows and walkers via postMessage (). Modules declare import and export like ES6 module.
- Memory: A size-adjustable Array Buffer, which is a linear array of bytes to be read by the low-level memory access command of the web assembly.
- Table: Size adjustment of references that cannot be stored in memory with raw bytes (for safety and portability *etc.*,) is possible, and the format is specified array.
- Instance: A pair of all the states used by the module and its modules.

It will also be thought of as a JavaScript function for efficiently creating high performance functions for Web Assembly because we could completely control the sequence of downloading, compiling, and running web assembly code with JavaScript. In the future, the web assembly module will be able to load like the E S6 module (using <script type = "module">).

The web assembly code can be executed at nearly native speeds on various types of platforms utilizing the functions provided by common hardware. Also, it is a low level languages have a text format at a level that allows them to create, display and debug by hand, that is, a level that people can read well. And it is designed to be able to safely return in a sandboxed execution environment, as well as other code on the web, web assembly code is also affected by the same source and permission policy of the browser [11].

Finally, the web assembly was designed to be used without friction with other web technologies, and to be able to manage backward compatibility. It can be utilized in many fields requiring 3D gaming, virtual / augmented reality, image processing, image / video editing, and other native performance using the web assembly. Therefore, in this paper, we try to analyze the domain of each technology in application development through web assembly and JavaScript, performance analysis of native code.

3. Analysis

3.1. Native Language vs Web Assembly

Web Assembly allow existing native languages to work in the web environment. It compiles native code in assembly language and then scripts that binary code. Therefore, we try to compare the performance of the web assembly and the existing native language. We let Unity's Angry-Bots run in the local environment and browser. The execution result is the same as in Figure 1.

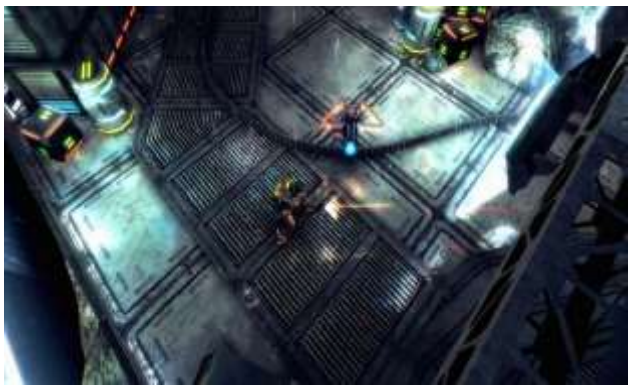

Figure 1. Unity Angry-Bots

As a method of converting web assembly code, there is a method of using Binaryen, Emscripten. Binaryen changes the code created in asm.js, and Emscripten converts C / C ++ code. The result of comparing the number of frames per second and the performance of the web assembly of the local environment's native language and the browser environment is the same as 2 in the following figure.

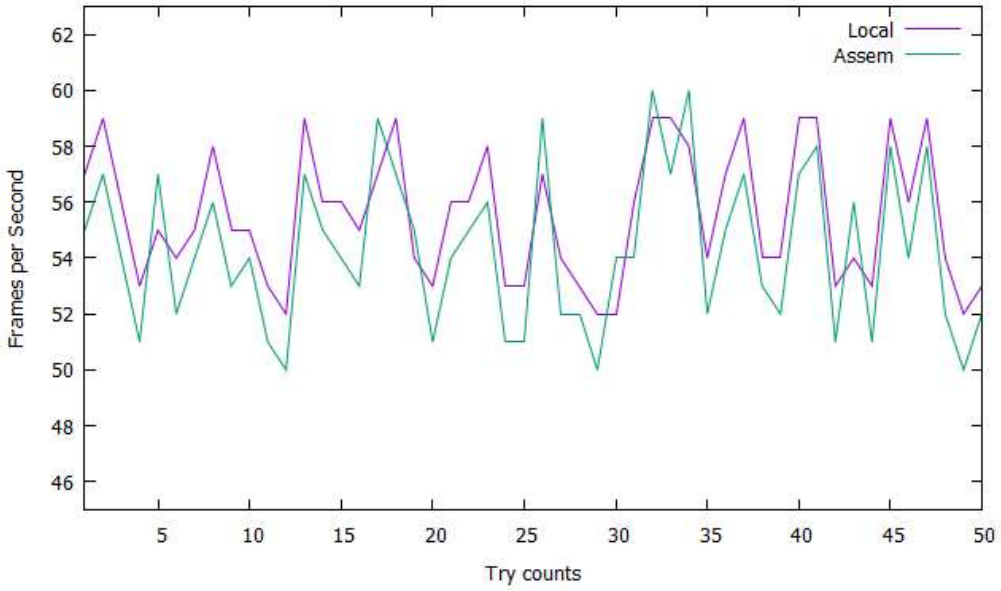

Figure 2. Analysis of Performance (Frames per Seconds)

Seems at Figure. 2, it can be confirmed that the performance of the result of operating in the local environment and the web assembly comes out at the same level. Performance of 50 to 60 frames per second is detected, which is the level of performance suitable for the user to use. It is let's know that the web assembly is not meant to replace the existing web language. The web assembly is a tool to complement existing ones. Through the analysis result, it can be judged that the web assembly is a complementary level language, not replacing the existing language.

3.2. JavaScript vs Web Assembly

The processing process of the job of the JavaScript engine which analyzed the performance difference between JavaScript and web assembly for comparison is as shown in Figure 3.

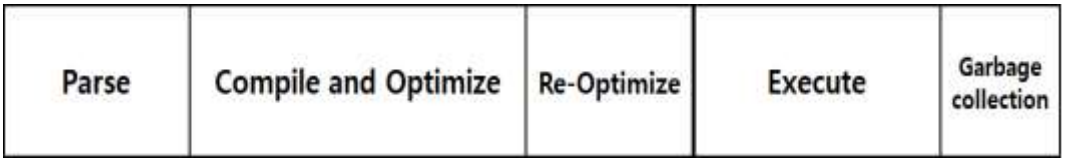

Figure 3. Work Process of JavaScript Engine

In Figure 3 above, the size of each Kahn means the approximate time it takes to execute the task. The Parse means the time it takes to process the source code into a form that the interpreter can execute. Compile and Optimize means the time taken by the baseline compiler and the optimizing compiler. Partly optimizing compiler work was excluded from the analysis, as it will not proceed in the main thread. Re-optimizing implies the time it takes for JIT (Just in time Compiler) to re-adjust when their home is wrong, again including all work to optimize or return to basic code. Execute means the time it takes to execute the code, and the Garbage Collection is the time it takes to free up unnecessary memory. These tasks are not done in a specific order and are proceeded as synchronized with one another. By running it as if it were decomposed into smaller units, we were able to improve the performance of the initial Java script. Web Assembly is often rated faster than JavaScript in many cases. The reason is as follows.

- Web assembly code takes less time to obtain code for simplicity than Java script.
- It takes less time to parse the syntax of JavaScript to decode the web assembly code.
- The web assembly code is closer to the machine code than the Java script code and pre-optimized on the server, so it takes less time to compile and optimize.
- Since web assemblies have pre-built types and other information, there is no need for the JavaScript engine to analyze at run time and no re-optimization time is needed.
- For web assembly code, developers need not know in advance for factors that slow down the compiler for performance.
- Because the web assembly code has a command set suitable for the machine, the execution time also takes less, mainly.
- Because it manages memory directly, garbage collection is not necessary.

In this way, web assemblies were developed in a way to shorten the time of traditional Java scripts and to develop better performance applications. In this paper, we try to analyze the performance of existing JavaScript and web assembly code for the same task. The analysis target is computation and loop, mainly used for application development, DOM navigation function. The majority of application performance can be divided into multimedia data control, simple iteration operation, object management, server data communication, and network environment. Therefore, this paper precedes analysis for simple iterative computation and object management, and continues to conduct further research thereafter. In order to execute simple iterative computation and analysis for object management, we implemented each function in Java script and Web assembly and analyzed execution speed. The Figure 4. Shows the JavaScript-based source code of each function.

```
// Test one source code : loop
var tmpArray = [];
var cnt = tmpArray.length;
for(i=0; i<cnt; i++){
    //Input the data to Array
    tmpArray[i] = i+1;
}

// Test two source code : operation
var tmpArray = [];
for (i=0; i<1000000; i++) {
    tmpArray.push('prefix', i, 'suffix');
}
var str = tmpArray.join('');

// Test three source code : Dom detection and add child
var tmp = document.createElement('tmp');
var tmpArray = [];
for (i=0; i<1000; i++) {
    tmpArray.push('<li>', '</li>');
}
tmp.innerHTML = tmpArray.join('');
document.body.appendChild(tmp);
```

Figure 4. JavaScript-based Code for Analysis

In the case of a loop, it analyzes the access to the index of the array through the loop and the operation of changing the data, in the case of operation, the operation of arithmetically operating the data in the array through a loop, in the case of DOM navigation, creates a new child node and, for jobs that attach that node to the existing tree

structure, Analysis was carried out. We created source code in JavaScript version and Web Assembly version, respectively, and operated on Chrome browser to calculate execution speed. Figures 5, 6, and 7 show the results.

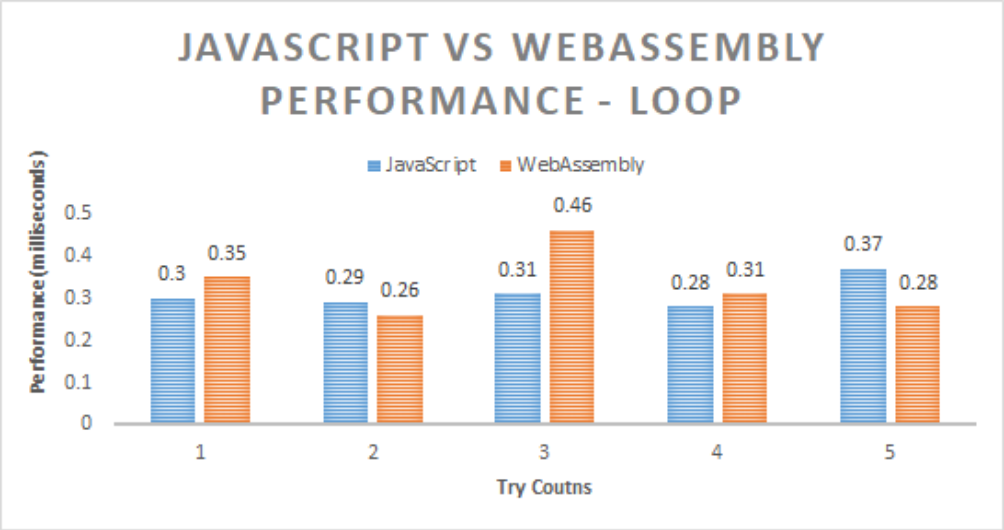

Figure 5. Result of Loop Analysis

Figure 5 shows the comparison of JavaScript and Web Assembly for loops. In the case of loops, the performance of JavaScript and Web Assembly did not differ substantially, on average, JavaScript 0.31ms, Web Assembly 0.332ms was measured. The loop does not utilize special resources, it is judged that there is no difference between the two languages because it is performance within the logic of the code.

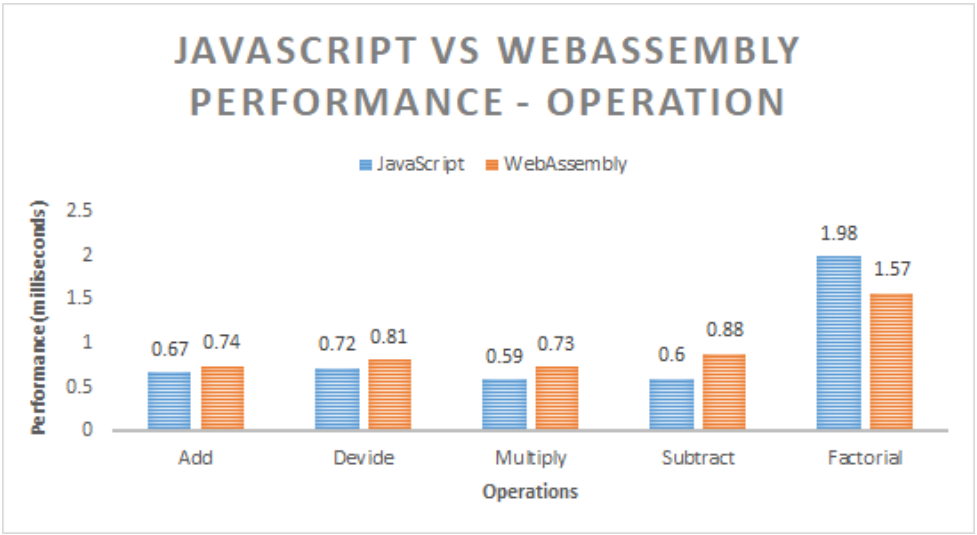

Figure 6. Operator Analysis Result

Figure 6 shows the comparison result between JavaScript and Web Assembly for operation. Like the loop statement, both results do not differ greatly. For factorial calculation only, the native language of the Web assembly has a high performance, but it is judged that it is not so much as to greatly affect the performance of the application. The average speed of arithmetic processing was measured with JavaScript 0.912ms, web assembly 0.946ms. Figure 7 shows the results of analysis for DOM navigation and creation of child nodes.

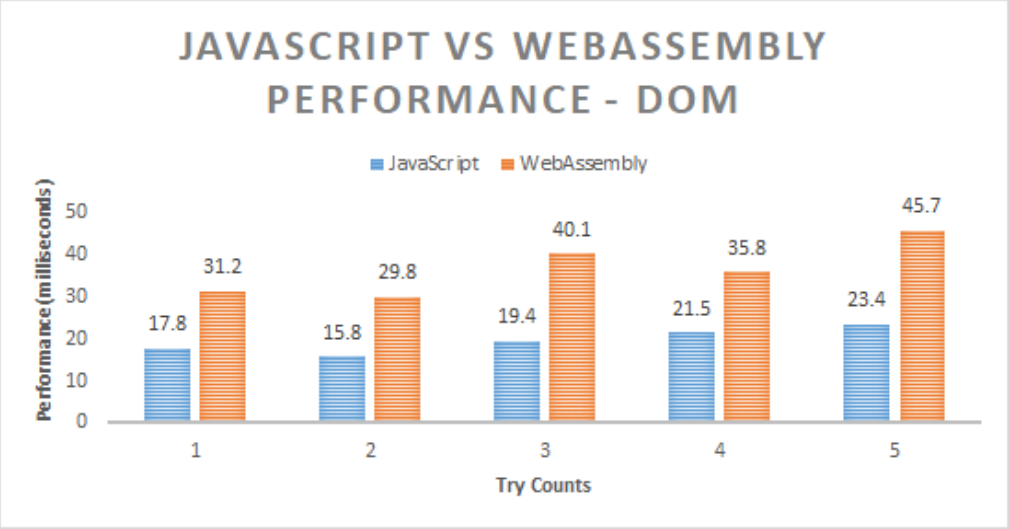

Figure 7. Result of DOM Navigation Creation and Analysis

In the case of DOM navigation and creation, unlike the two analytical results that went forward, JavaScript performance showed excellent. This means that JavaScript is not a native language, but is effective for tasks that perform existing HTML structure changes. On average Java scripts showed faster performance of about 20ms than native languages.

Table 1 shows a comprehensive analysis result for simple iterative calculation and object management.

Table 1. Performance Integrated Analysis

		JavaScript Performance(ms)	Web Assembly Performance(ms)
Loop		0.31	0.332
Dom		19.58	36.52
Operation	**ADD**	0.67	0.74
	DEVIDE	0.72	0.81
	MULTIPLY	0.59	0.73
	SUBTRACT	0.6	0.88
	FACTORIAL	1.98	1.57

The JavaScript and Web Assembly are mutually complementary, as can be seen through the performance comparison of each task. Also, the function compatible with HTML is a Java script, and it can be judged that developing functions in utilizing computing power in a native language is more efficient.

4. Conclusion

In the web environment, it is changing to a form that expresses the virtual world with simple information provision and requires voluntary participation of users. Such a change makes it possible to apply the Internet, artificial intelligence technology of the fourth industrial revolution to various fields.

However, existing Web technologies are often inadequate to utilize the performance of hardware compared to native languages. To compensate for these drawbacks, a new type of code called Web Assembly was developed, which allows existing native languages to run on web virtual machines. This makes it possible to perform native level performance even in the web environment, and it has become possible to use it in various fields.

It is unlikely that the web assembly is a technology that came out to replace JavaScript running on an existing web. Web assemblies are right to operate on the web, control various resources and look at what is supplementing JavaScript.

For efficient mutual complementation, we will maximize the advantages of each language and design algorithms and logic accordingly. Therefore, in this paper, we conducted a simple iterative work which is mainly used for application development, which affects application performance, and performance analysis of DOM navigation and creation. As a result, it was analyzed that performance of JavaScript is better in the portion that utilizes computing resources, the performance of Web assembly was good, and the part that executes the task of existing HTML structure. The other two languages in basic arithmetic processing did not show a big difference.

Analysis of Web application performance will be done deeper considering network, limited resource environment, *etc*. Therefore, this research team will conduct research on simplifying the assembly code and integrating it with the advantages of existing Web technology in the future.

Acknowledgments

This research was financially supported by the Ministry of SMEs and Startups and Korea Institute for Advancement of Technology (KIAT) through the Research and Development for Regional Industry.

This paper is a revised and expanded version of a paper entitled [Analysis Performance of Web Assembly based on JavaScript Engine for H/W Acceleration] presented at [AWITC, Busan, and February 9, 2018].

References

[1] D. Guinard and V. Trifa, “Towards the web of things: Web mashups for embedded devices”, Workshop on Mashups, Enterprise Mashups and Lightweight Composition on the Web (MEM 2009), in proceedings of WWW (International World Wide Web Conferences), Madrid, Spain. **(2009)**.

[2] N. Van Es, “A performant scheme interpreter in asm. Js”, Proceedings of the 31st Annual ACM Symposium on Applied Computing. ACM, **(2016)**.

[3] https://developer.mozilla.org/ko/docs/WebAssembly.

[4] https://www.w3.org/community/webassembly.

[5] M. Reiser and L. Bläser, “Accelerate JavaScript applications by cross-compiling to Web Assembly”, “Proceedings of the 9th ACM SIGPLAN International Workshop on Virtual Machines and Intermediate Languages. ACM, **(2017)**.

[6] S. Y. Park, J. Chang and T. Kihl, “Document classification model using Web documents for balancing training corpus size per category”, Journal of information and communication convergence engineering, vol. 11, no. 4, **(2013)**, pp. 268-273.

[7] A. Haas, “Bringing the web up to speed with Web Assembly”, Proceedings of the 38th ACM SIGPLAN Conference on Programming Language Design and Implementation, ACM, **(2017)**, pp. 185-200.

[8] S. Letz, Y. Orlarey and D. Fober, “Compiling Faust audio DSP code to Web Assembly”, **(2017)**.

[9] J. Renner, S. Cauligi and D. Stefan, “Constant-Time Web Assembly”.

[10] H. Seo and H. Kim, “Low-power encryption algorithm block cipher in JavaScript”, Journal of information and communication convergence engineering, vol. 12, no. 4, **(2014)**, pp. 252-256.

[11] C. Watt, “Mechanising and verifying the Web Assembly specification”, Proceedings of the 7th ACM SIGPLAN International Conference on Certified Programs and Proofs. ACM, **(2018)**.

Authors

Park Jin-Tae, he received the B.S., M.S degree in Computer Engineering from Korea University of Technology and Education, Korea. He is currently a Ph.D. course at the same university. His main research interest is mobile network technology and IoT, Web Script Engine.

Email: wlsxo05@koreatech.ac.kr

Kim Hyun-Gook, he is currently a B.S course student at the Computer Engineering from Korea University of Technology and Education, Korea. His main research interest is mobile network technology.

Email: hy1392@koreatech.ac.kr

Moon Il-Young, he received his B.S., M.S., and Ph.D. degrees from the Department of Information Telecommunication, Korea Aerospace University, in 2000, 2002, and 2005, respectively. He was a senior researcher at the Korea Agency for Digital Opportunity and Promotion in 2005. He is presently working as an associate professor at the Korea University of Technology and Education. His research interests include wireless networks and mobile internet.

Email: iymoon@koreatech.ac.kr