*Advances in Science, Technology and Engineering Systems Journal*
*Vol. 7, No. 5, 53-61 (2022)*
www.astesj.com
*Special Issue on Innovation in Computing, Engineering Science & Technology*

**ASTES Journal**
**ISSN: 2415-6698**

# A Proposal of Code Modification Problem for Self-study of Web Client Programming Using JavaScript

Khaing Hsu Wai[1], Nobuo Funabiki[*,1], Khin Thet Mon[1], May Zin Htun[1], San Hay Mar Shwe[1], Htoo Htoo Sandi Kyaw[2], Wen-Chung Kao[3]

[1]*Department of Information and Communication Systems, Okayama University, Okayama, Japan*

[2]*Department of Computer and Information Science, Tokyo University of Agriculture and Technology, Tokyo, Japan*

[3]*Department of Electrical Engineering, National Taiwan Normal University, Taipei, Taiwan*

ARTICLE INFO

*Article history:*
*Received: 16 May, 2022*
*Accepted: 05 September, 2022*
*Online: 24 September, 2022*

*Keywords:*
*web client programming*
*JavaScript*
*HTML*
*CSS*
*code modification*
*coding reading*
*self-study*

ABSTRACT

In current societies, *web application systems* take central roles in computer systems. Thus, *web client programming using JavaScript* has increased values to add dynamic features and functions in web pages by well working with *HTML* and *CSS*. In this paper, as a new type of exercise problem for its self-study, we propose a *code modification problem (CMP)*. In CMP, a source code with *HTML/CSS* elements and *JavaScript* functions for study, and the screenshots of both the original and the slightly altered web pages are provided to the students, who will need to edit the source code to generate the modified page. The goal of CMP is for students to carefully read the source code and comprehend how to use the components and functions through modifying parameters, values, or messages there. *String matching* is used to check the correctness of any answer. Through solving CMP instances, the students are expected to master the basic concepts of *web client programming*. To evaluate the proposal, we generated and assigned 25 CMP instances to 37 students in Okayama University. In addition, we offered project assignments of freely implementing source codes by referring to solved CMP instances to evaluate their learning effects. With the solution results, the validity of the proposal has been confirmed.

## 1 Introduction

In current societies, computer systems using software and hardware have big impacts on our daily lives. They are crucial to enhancing the capabilities of communications, calculations, and information services. By executing a lot of effective programs that have been created using various programming languages, computer systems are illustrating their powers. In computer systems, several types of programming languages have been used together, depending on the fields and applications of them. Thus, studying the commonly used plural programming languages is essential to develop effective and efficient computer systems for the students in departments of computer science (CS) or information technology (IT) in academic institutions. Especially, web application systems have taken prominent roles in computer systems with expansions of the Internet. For this reason, learning *web client programming with JavaScript* is crucial for students majoring in IT or CS.

The primary scripting language running on web browsers is *JavaScript*, which is sometimes referred as *JS*. *JavaScript* is crucial to modern *web application systems* [1]. Currently, *JavaScript* is adopted in 97% of websites in the world, making it the most popular client-side scripting language for *web client programming*. By combining the common web technology by the *Hyper Text Markup Language (HTML)* and *Cascading Style Sheets (CSS)* , *JavaScript* can provide dynamic features on a *web page*. The structure and meaning of the page are provided in *HTML*. The layout, background colors, and fonts used in *HTML* content are described in *CSS*. Then, the *document object model (DOM)* is used for *JavaScript* interactions with them.

In order to learn a new programming language, novice students should start their study by solving simple and short exercise problems to gain the basic understanding of the grammar and key concepts for *code reading study*. If they cannot read and understand simple source codes, they are unable to properly write them by themselves for *code writing study*. This step-by-step programming learning is important for novice students. After well completing *code reading study*, novice students should move to *code writing study*. To offer step-by-step self-study environments of programming languages to novice students, the project of developing the web-based *programming learning assistant system (PLAS)* has been

[*]Corresponding Author: Nobuo Funabiki, 3-1-1 Tsushimanaka, Okayama University, funabiki@okayama-u.ac.jp

established. Currently, well-known programming languages of C, C++, Java, Python, and JavaScript are supported in PLAS. PLAS offers different types of exercise problems with different learning goals and difficulty levels for the step-by-step studies, as the learning materials or contents on the common system platform.

In PLAS, there exist five types of exercise problems for code reading/writing studies, namely, *grammar-concept understanding problem (GUP)*, *value trace problem (VTP)*, *element fill-in-blank problem (EFP)*, *code completion problem (CCP)*, and *code writing problem (CWP)*.

The *grammar-concept understanding problem (GUP)* [2] offers questions about the source code's keywords by defining terms like reserved words and common libraries in programming languages. The *value trace problem (VTP)* [3], [4] requests the output values of significant variables and output messages in the provided source code. The *element fill-in-blank problem (EFP)* [5], [6] asks the students to complete the missing elements in the provided source code by comprehending its syntax and semantics in order to discover the original source code. The *code completion problem (CCP)* [7], which is intended for debugging study, requests to correct and complete the provided source code when there are blank or incorrect elements in the source code. The *code writing problem (CWP)* [8] requests to complete a new source code from scratch that can pass the given test code. The correctness of the student's answer is automatically checked by *string matching* with the correct one for *GUP*, *VTP*, *EFP*, and *CCP*, and by *software testing* using test codes for *CWP*.

In PLAS, students should first solve the problems related to *code reading study*. They include *GUP* and *VTP* to understand keyword definitions and source code control flows. Next, to learn how to complete source codes by themselves, they should solve *EFP*, *CCP*, and *CWP* in this order. Then, novice students will possibly continue programming studies without dropping out, while solving the problems in PLAS step-by-step, and resulting in improved code reading and writing skills.

However, due to the fact that most web pages are written with the combination of *JavaScript*, *HTML*, and *CSS*, any type of exercise problem currently available in PLAS may not be suitable for studying *web client programming*. After studying each language separately, students must be able to relate them in the source code in *web client programming*.

In this paper, we propose a *code modification problem (CMP)* as the new exercise problem type in PLAS for effective self-study of *web client programming using JavaScript*. Since one web page is usually generated by combining the layout styles of HTML/CSS elements with JavaScript library functions, the effective way to learn them is to read and understand the sample source codes where they are included. A *CMP* instance includes a source code with the HTML/CSS elements and JavaScript functions that should be studied, as well as the two screenshots of the originally produced web page and another modified web page. Then, the original source code is requested to be modified so as to generate the second web page. This CMP is designed for students to carefully study the source code and understand the use of the elements/functions, while changing some parameters, values, or messages. *String matching* is used to check the correctness of any answer. It is expected that they can obtain the basic concepts of *web client programming* through solving the given CMP instances.

To evaluate the proposal, we generated and assigned 25 CMP instances to 37 students in Okayama University. Moreover, we offered project assignments that allow students to implement source codes freely by referring to solved CMP instances to evaluate their learning effects. The validity of the proposal has been confirmed by their solution results.

The outlines of this paper is organized as follows: Section 2 discusses related works in literature. Section 3 proposes the *code modification problem (CMP)*. Section 4 evaluates the analysis results of CMP. Section 5 discusses the learning effects and learning validation after solving the CMP instances. Finally, Section 6 concludes this paper with future works.

## 2 Related Works in Literature

In this section, we discuss related works to this study in literature.

In [9], the authors proposed an educational game approach called *Reduct* to instruct novice students on fundamental JavaScript programming principles, such as functions, Booleans, equivalence, conditional expressions, and mapping functions onto sets. The designs used theories of progression design and skill learning to scaffold concepts and motivate players to create accurate mental models of the codes. The current objective of this paper is to teach up to the level of advanced and fundamental functional programming in JavaScript.

In [10], the authors described a prototype system to aid students in learning the web language JavaScript. They discussed how portable intelligent exercises activities were implemented and tested them in the web programming course. According to survey assessment data, they demonstrated that the system may assist students in learning of JavaScript, the web-based programming language.

In [11], the authors proposed the *JavaScript development environment (JDE)* for the purpose of supporting programming education. The JDE offers a setting in which JavaScript programming may be done anywhere, at any time. Additionally, the JDE may offer comprehensive snippet features that make it possible to write code using a limited number of actions. It is appropriate for usage on smartphones because it is an environment based on a browser. The JDE can edit HTML, CSS, and JavaScript-enabled web pages.

In [12], the authors presented contributions of an interactive e-learning course for teaching web technologies including JavaScript and HTML as a component of the Moodle platform. This course may have been taught using traditional methods of instruction, or an alternative, utilizing the *Scrum* agile software development methodology and the *EduScrum* teaching methodology.

In [13], the authors proposed an interactive serious programming game for JavaScript programming course at their university. The gamification pattern-based method was used to create this game, together with the *Technology Acceptance Model (TAM)*, and the *Technology-Enhanced Training Effectiveness Model (TETEM)*. Using pre-test and post-test knowledge evaluations, *TAM*, and *TETEM*, they presented the game's evaluation findings.

In [14], the authors made comparative descriptions of several online platforms for teaching programming and chose engaging assignments from the site used to educate students named *hackerrank.com*. They investigated user experiences with *online coding* *platforms (OCP)* and contrasted the features of various online platforms that should be utilized to teach programming to aspiring computer scientists and programmers via distance learning. In addition, they also suggested using online programming simulators to enhance computer science instruction, taking into account functionality, as well as students' preparation levels and expected results of learning.

In [15], the authors proposed a game-based learning environment to assist beginner students learning programming. It uses game creation tasks to make basic programming easier to understand and includes idea visualization approaches to let students manipulate game objects to learn important programming concepts.

In [16], in order to investigate students' perceptions of this learning environment, the authors developed a collaborative, learning environment based on the problems powered by the technology for dynamic web. The research was planned as a qualitative study. A interview format that was semi-structured was created to get the opinions of the students about the learning environment, which was supported by technologies for dynamic web and used collaboratively solving issues techniques. Their findings imply that collaborative learning techniques focused on problems and the learning environment at a community college can benefit from technologies for dynamic web pages.

## 3 Proposal of Code Modification Problem

In this section, the *code modification problem (CMP)* for self-study of *JavaScript* based *web client programming* in PLAS is discussed.

### 3.1 Definition of Code Modification Problem

In a CMP instance, a source code containing the HTML/CSS elements and the functions to be studied here, and a pair of the screenshot of the original web page generated by the code and the screenshot of the slightly altered web page are provided to students. These web pages could have different parameters, functionalities, or variables. Then, the students are requested to modify the source code to generate the altered web page. By solving the given CMP instances, it is expected that the students can master the basic concepts of *web client programming* and understand the interacted use of HTML, CSS, and JavaScript in the source code. *String matching* is used to check the correctness of any answer in the source code.

CMP is designed with the following goals:

1. Source codes of various kinds are provided along with full forms in order to assist novice students with learning *web client programming using JavaScript*.
2. By answering the questions in CMP instances correctly, students can learn how to generate web pages using various JavaScript functions.
3. As a result of using *string matching*, the student feedback is immediately provided when answers are automatically marked.
4. A novice student who has never studied *web client programming* can answer the questions without encountering serious difficulties.

### 3.2 Generation Procedure of CMP Instance

The following procedure can be used to generate a new CMP instance:

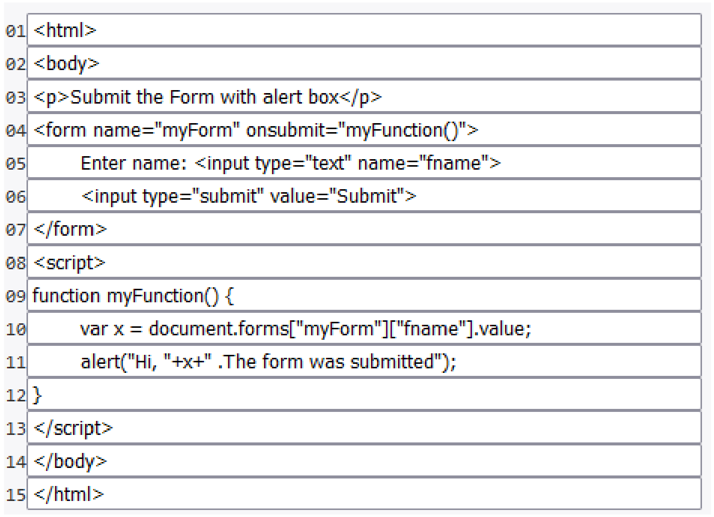
1. From a website or book that contains the library functions to be examined in this instance, choose a proper source code with HTML, CSS, and JavaScript to develop the web page.
2. Save the screenshot of the web page after running the source code in a web browser.
3. Identify and determine the parts of the source code that students should modify to better understand the intended functionality. Currently, this step is manually carried out. The automatic processing will be studied in future works.
4. Save the screenshot of the altered web page after running the modified source code in a web browser.
5. Due to the fact that HTML tags are not visible on web browsers, replace the HTML tags with the HTML entities according to their numbers and names.
6. Make the input text file for the newly created CMP instance that consists of the problem statement, the original source code, and the modified source code as the correct solution.
7. Using this text file, generate the CMP instance files, which are combined with the HTML, CSS, and JavaScript, for the answer interface on a web browser using the *instance generation program*. It should be noted that this program has already been implemented and used for GUP, VTP, EFP, and CCP.
8. Insert the screenshots of the original and modified web pages into the HTML file for the CMP instance.

### 3.3 Example of Generating CMP Instance

Next, an example web page source code will be used to discuss the details of the CMP instance generation.

#### 3.3.1 Original Source Code

The original source code should include the fundamental JavaScript library functions for *web client programming* to be studied in this instance. The example source code in Figure 1 shows the web page that displays the *alert box* after clicking *Submit button* from accepting the user name in the input form.

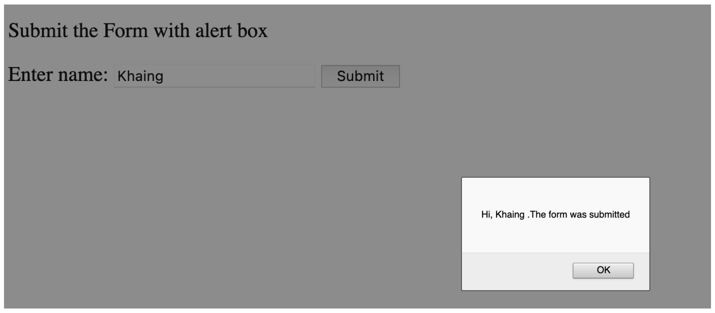


```
<html>
<body>
<p>Submit the Form with alert box</p>
<form name="myForm" onsubmit="myFunction()">
        Enter name: <input type="text" name="fname">
        <input type="submit" value="Submit">
</form>
<script>
function myFunction() {
        var x = document.forms["myForm"]["fname"].value;
        alert("Hi, "+x+" .The form was submitted");
}
</script>
</body>
</html>
```

Figure 1: Original source code for CMP instance #19.

### 3.3.2 Original Web Page

The screenshot of the original source code for CMP instance #19 in Figure 2 shows the web page that was created using the source code in Figure 1. This generated web page is offered as a reference for understanding the source code.



Submit the Form with alert box

Enter name: Khaing

Submit

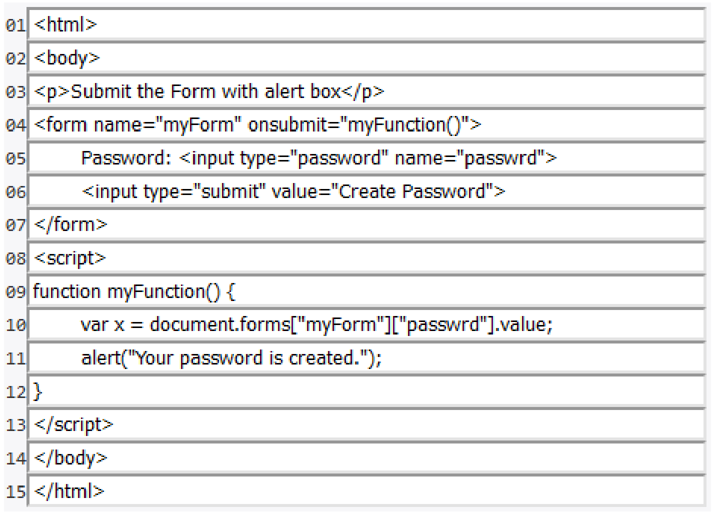
Hi, Khaing .The form was submitted

OK

Figure 2: Web page by source code for CMP instance #19.

### 3.3.3 Modified Source Code

The original source code can be transformed into the modified source code by changing a few parameters and functions. It is important for students to understand the connections among the corresponding elements of HTML, CSS, and JavaScript when creating web pages. Figure 3 displays the source code for the correct answer. In this example instance, it is necessary to change the text messages in the alert box and the button, as well as the type and the name in the "input" tag.

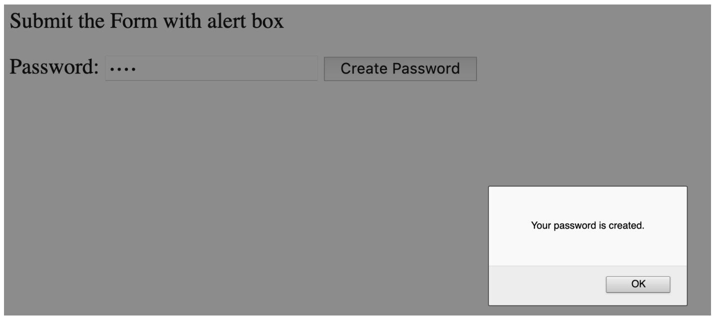


```
<html>
<body>
<p>Submit the Form with alert box</p>
<form name="myForm" onsubmit="myFunction()">
        Password: <input type="password" name="passwrd">
        <input type="submit" value="Create Password">
</form>
<script>
function myFunction() {
        var x = document.forms["myForm"]["passwrd"].value;
        alert("Your password is created.");
}
</script>
</body>
</html>
```

Figure 3: Modified source code for CMP instance #19.

### 3.3.4 Modified Web Page

Figure 4 shows the screenshot of the altered web page by the modified source code.



Submit the Form with alert box

Password: ••••

Create Password

Your password is created.

OK

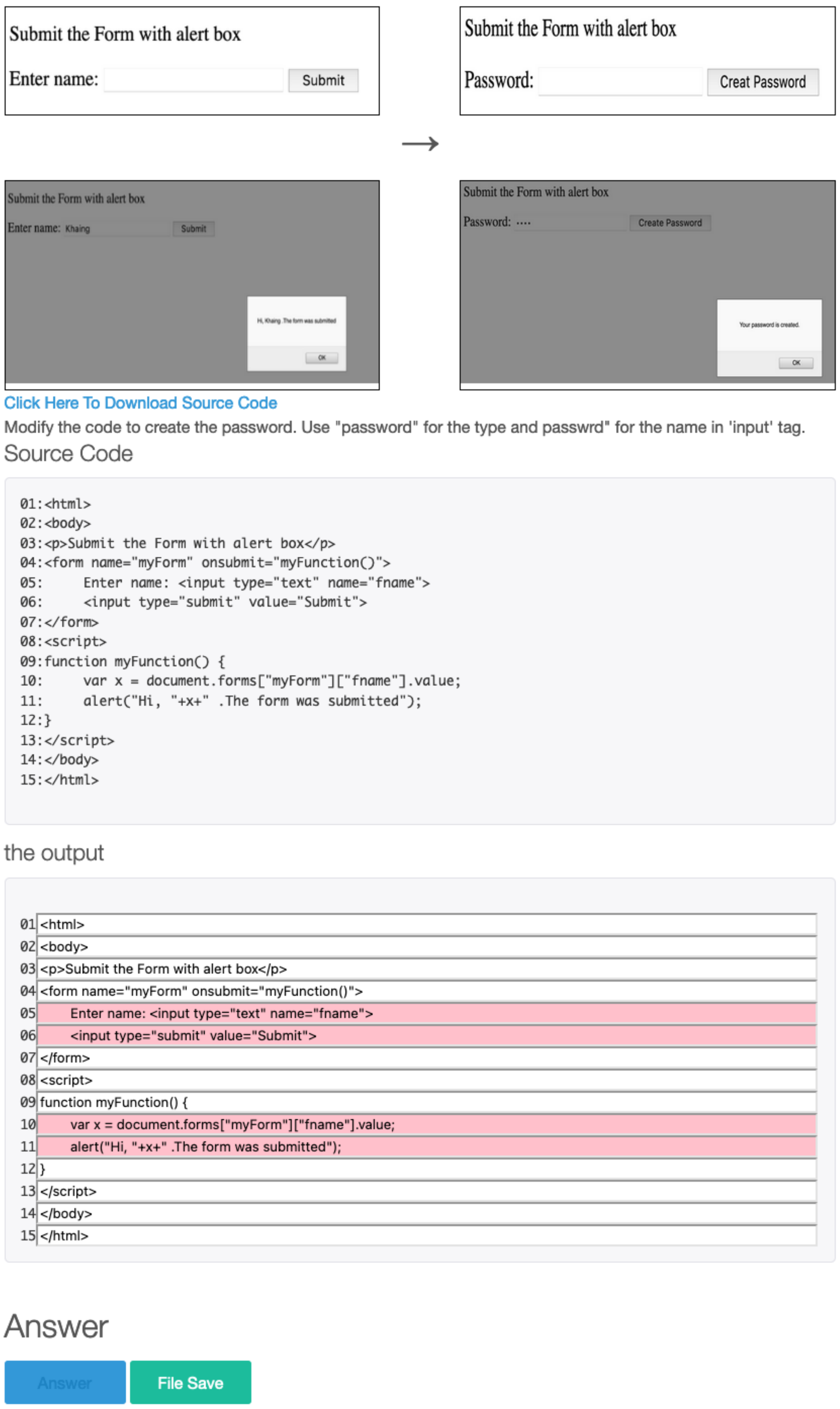
Figure 4: Modified web page for CMP instance #19.

## 3.4 Answer Interface

The answer interface is implemented using HTML, CSS, and JavaScript, where the JavaScript program handles the answer marking for the CMP instance. This allows both the online use and the offline use of the answer interface. In order to prevent students from cheating, the correct answers in any CMP instance are encrypted using *SHA256* [7].

Figure 5 illustrates the response interface for the CMP instance that requests the change of the form for the alert box. The source code lines in the input forms, which are indicated by red backgrounds, must be modified by students to complete this CMP instance. To better understand the source code, students can also download and run the source code, if they want to see how the web page is created and how it works.

After completing all the required changes to the input fields, the student can click the "Answer" button to confirm the validity of the answers. The relevant input form's background color is changed to *red* if the answer on the line is incorrect. Otherwise, it turns *white*. The student can submit their answers again until every answer is verified as correct.


Submit the Form with alert box
Enter name:
Submit
Submit the Form with alert box
Password:
Creat Password
Submit the Form with alert box
Enter name: Khaing
Submit
Submit the Form with alert box
Password: ····
Create Password
OK
OK

Hi, Khaing .The form was submitted

Your password is created.

Click Here To Download Source Code

Modify the code to create the password. Use "password" for the type and passwrd" for the name in 'input' tag.

Source Code

```
<html>
<body>
<p>Submit the Form with alert box</p>
<form name="myForm" onsubmit="myFunction()">
    Enter name: <input type="text" name="fname">
    <input type="submit" value="Submit">
</form>
<script>
function myFunction() {
    var x = document.forms["myForm"]["fname"].value;
    alert("Hi, "+x+" .The form was submitted");
}
</script>
</body>
</html>
```

the output

```
<html>
<body>
<p>Submit the Form with alert box</p>
<form name="myForm" onsubmit="myFunction()">
    Enter name: <input type="text" name="fname">
    <input type="submit" value="Submit">
</form>
<script>
function myFunction() {
    var x = document.forms["myForm"]["fname"].value;
    alert("Hi, "+x+" .The form was submitted");
}
</script>
</body>
</html>
```

Answer

Answer File Save

Figure 5: Answer interface for CMP instance #19.

## 4 Evaluation

This section evaluates the proposed code modification problem (CMP) for *web client programming using JavaScript*.

### 4.1 Evaluation of CMP Instances

In this evaluation, 25 CMP instances are generated, to examine fundamental concepts and functions in *web client programming* for performing dynamic behaviors of web pages using JavaScript, HTML, and CSS. The topic/function, the number of lines in the source code, and the number of elements that need to be modified for each CMP instance are displayed in Table 1. Other topics like using media devices will be studied in our upcoming works.

### 4.2 Solution Results

A total of 37 first-year master students in Okayama University, who have not taken any formal course in JavaScript programming, are assigned these 25 CMP instances. Prior to this assignment, we only offered a few websites as references to them.

#### 4.2.1 Results of Individual Students

The results of the 37 individual students are first analyzed in our evaluations. The correct answer percentage (%) and the number of submissions in average among the 25 CMP instances are shown for each student in Figure 6.

According to the results, out of the 37 students, 22 students correctly answered the question and achieved the 100% correct rate, and 10 students scored above 90%. Only five students could not reach 90%. Among these five students, three are considered giving up answering the problems as zero answer submission attempts, and two students had not solved and submitted all the problems. Therefore, in general, the novice students can solve the generated CMP instances through self-study of *web client programming using JavaScript*.

Since each instance requires at least one submission of the answer, 25 is the least number to solve the 25 CMP instances. There are 131.4 submission times on average. These students generally submitted answers 2 – 14 times to solve one CMP instance. As a result, they carefully prepared and checked their answers before submitting them.

#### 4.2.2 Results of Individual Instances

Next, we examine the solution results of individual 25 CMP instances. The correct answer percentage (%) and the number of submissions on average are shown for each of the 25 CMP instances among the 37 students in Figure 7.

According to the results, the instance at ID=18 gets the lowest correct rate, which is 90.8%, and the instance at ID=7 achieves the highest correct rate, which is 100%. The maximum number of submissions is 10 for the instances at ID=18 and ID=23 and the minimum ones are 3.3 for the CMP instances of the instances at ID=2, ID=17 and ID=19. Regarding the proportion of the correct answers, 30 elements/functions are needed to be changed in the instance at ID=18. In this case, we can assume that students looked at the screenshots rather than reading the instructions. Some students had not tried to solve this problem from the answer results. In contrast, the source code of the instance at ID=7 was simple for students where they just need to change the colors and the lists.

As for the number of submission, the instances at ID=2, ID=17, and ID=23 are simple to find the necessary modifications. On the other hand, the instances at ID=18 and ID=23 require several modifications. Thus, students submitted their answers 10 times on average. In general, the CMP instances are simple to be solved. However, some instances contain long source codes, where students took time to read and understand them, and made higher submission times.

## 5 Project Assignment for Learning Effect Evaluation

In this section, we present *project assignments* to evaluate learning effects of the *code modification problem (CMP)* in *JavaScript-based web-client programming*.

### 5.1 Overview

The proposed CMP is designed for students to read a source code carefully by asking changes of some parameters, values, or messages there, and to understand how to use the HTML/CSS elements and the JavaScript library functions appearing in the code for *web client programming*. However, the CMP does not ask writing source codes by students, which will be a weak point of this approach. Thus, we will evaluate learning effects in code writing to students after solving the CMP instances.

For this purpose, we prepared two *project assignments*, requesting to design web pages and complete the source codes by referring to some given CMP instances. We made the corresponding answer interface to the project assignment that contains the *problem statement* on the requirements, the *input form* for the answer source code, the *web page output area* by running the code, and the *hint* including the sample page layout, the related CMP instances, and the short instruction video. The web page generated by running the code is shown so that students can easily test their source codes. The project assignments were assigned to the same students after solving the 25 CMP instances.

### 5.2 Project Assignment #1

The first project assignment is *Timer* that requires to implement the text input form, the start, stop, and clear buttons, and the time counting funcion by using the interval function. Figure 8 illustrates the sample web page.

#### 5.2.1 Problem Statement

In this assignment, the problem statement is given by:
"Create and develop simple "Timer" using HTML, CSS and JavaScript. You timer will count the minutes by using *setInterval ( )* function. You can see the hints by clicking the Hint button below. You can also try to write and run codes in the 'Code' area and see how your codes work in the 'Output' area. If your code is OK, you can save your project with your student ID."

Table 1: Generated CMP instances.

| ID | topic/function | # of lines | | # of modified elements | |
|---|---|---|---|---|---|
| | | HTML/CSS | JavaScript | HTML/CSS | JavaScript |
| 1 | JavaScript object | 6 | 4 | 1 | 4 |
| 2 | JavaScript class | 13 | 10 | 2 | 2 |
| 3 | JavaScript math | 7 | 3 | 3 | 1 |
| 4 | click button | 10 | 3 | 2 | 1 |
| 5 | disable button | 12 | 20 | 2 | 3 |
| 6 | circle drawing | 5 | 14 | 0 | 5 |
| 7 | unordered list | 19 | 7 | 6 | 3 |
| 8 | radio button | 11 | 4 | 5 | 1 |
| 9 | checkbox | 8 | 12 | 1 | 1 |
| 10 | information and color change | 15 | 3 | 5 | 1 |
| 11 | color form type | 10 | 12 | 2 | 2 |
| 12 | element position change | 27 | 10 | 2 | 2 |
| 13 | rotating image | 14 | 7 | 1 | 3 |
| 14 | clickable image map | 23 | 5 | 3 | 1 |
| 15 | image button with counter | 22 | 7 | 3 | 3 |
| 16 | background image position change | 21 | 6 | 4 | 2 |
| 17 | text transformation from text area | 37 | 11 | 7 | 1 |
| 18 | file input type with alert box | 25 | 5 | 8 | 1 |
| 19 | input form with alert box | 9 | 6 | 2 | 2 |
| 20 | numbers addition/subtraction from input form | 33 | 8 | 3 | 2 |
| 21 | progress bar with clicking button | 23 | 23 | 4 | 3 |
| 22 | slider control with range input type | 39 | 9 | 10 | 1 |
| 23 | bar chart drawing from input value | 49 | 14 | 5 | 3 |
| 24 | table row insertion with button | 41 | 9 | 8 | 3 |
| 25 | table column deletion with button | 66 | 16 | 32 | 2 |

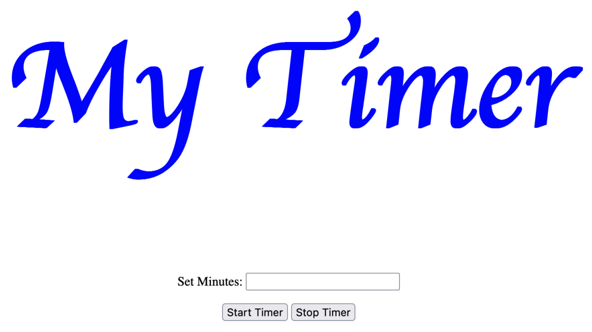


Figure 8: Sample *Timer* page for assignment #1.

### 5.2.2 *Hint*

In this assignment, the hints are given by:
"For *Timer*, you will need input form to set the minutes and two buttons: start and stop for setting the timer based on the user input value. You can also see Problem #19 and #21 for the reference.
For the *layout interface*, you can create your free design for simple minute count timer. In the following example video, the timer will start as soon as the user put the input minutes value and start the timer button. The timer will be stopped and clear the interval when the user click the stop timer button.
The *setInterval( )* method executes a function or evaluates an expression at specified intervals for *JavaScript library functions* (in milliseconds). Until *clearInterval()* is invoked or the window is closed, the *setInterval( )* method will keep running the procedure. The ID value returned by setInterval( ) is used as the parameter for the *clearInterval( )* method."

### 5.2.3 *Result and Discussion*

Among the 37 students, only seven students completed this project assignment by making the source codes satisfying the requirements on the interface and functions. Five students among them achieved the 100% correct rate in solving the 25 CMP instances.

Feedbacks from other students who could not complete this assignment suggested that they found difficulties in combined use of HTML, CSS, and JavaScript, which usually results in long source codes. They also found difficulties in debugging source codes, because the web browser does not show any error message while running them. The use of an advanced IDE returning the error messages should be recommended to edit the source code in *web-client programming*.

Then, it was observed that the web page designs of the submitted source codes are unique from each other. Some pages show the timer with minutes and seconds, while others additionally show hours, and even years, months, and days. It is concluded that these seven students well studied *web-client programming using JavaScript*.

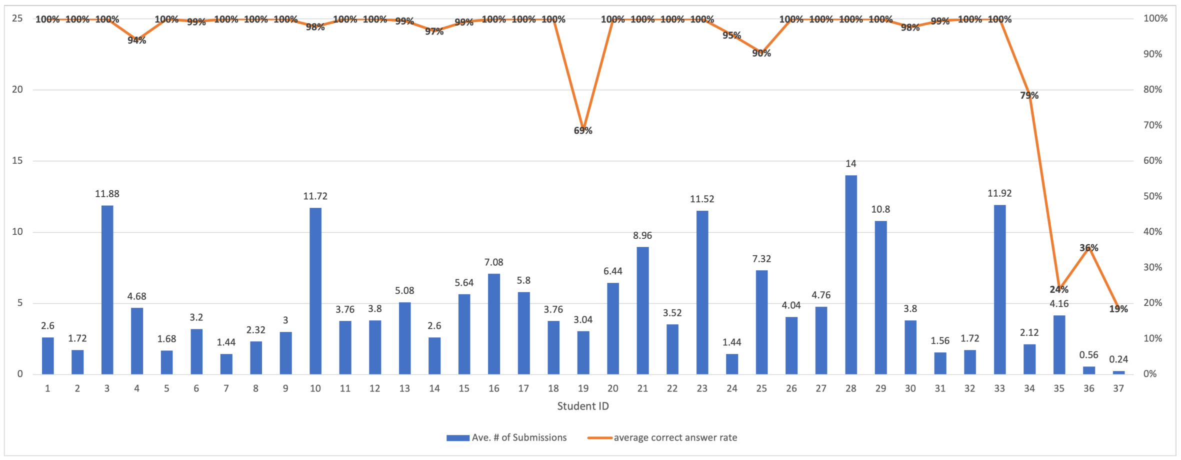


Figure 6: Results for each student.

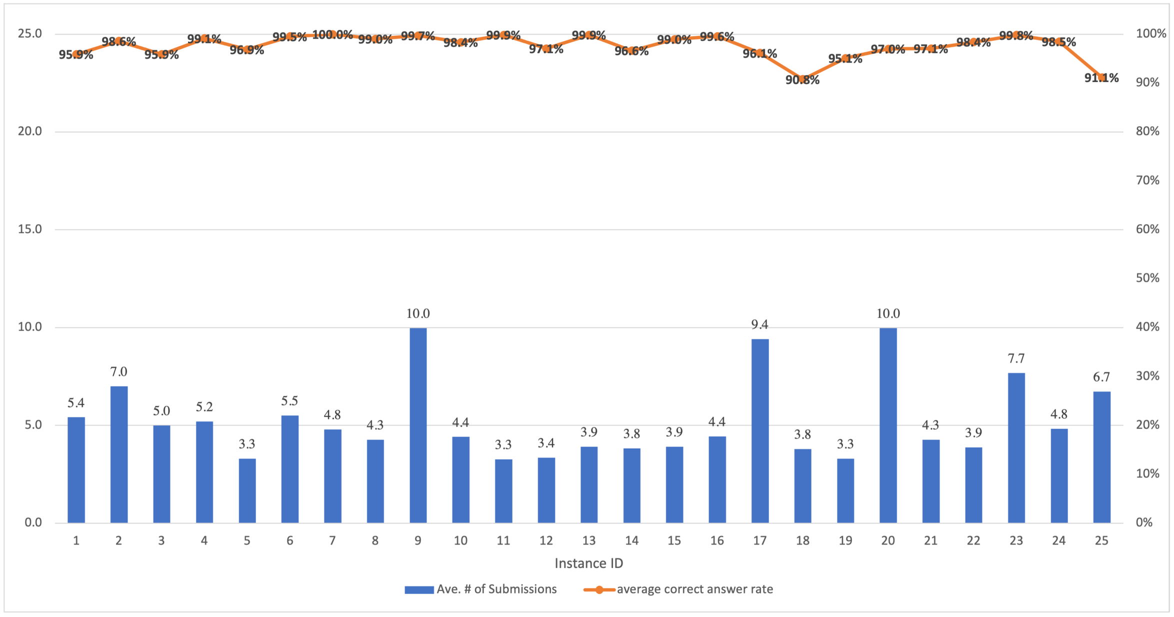


Figure 7: Solution results for individual CMP instances.

## 5.3 *Project Assignment #2*

The second project assignment is *Calculator* that requires to implement the table layout, the result area, the text input form, the calculation button, and the four arithmetic operations: addition, subtraction, multiplication, and division. Figure 9 illustrates the sample web page. Numbers (0-9), and operators (+, −, ∗, /, %, =) should appear in the table layout and the result area.

### 5.3.1 *Problem Statement*

In this assignment, the problem statement is given by:
"Create and develop simple "Calculator" using HTML, CSS, and JavaScript. Your calculator may contain numbers (0-9), simple basic calculations (+, −, ∗, /, %, =) and result area to show the calculation answers. You can see the hints by clicking the Hint button below. You can also try to write and run codes in the 'Code' area and see how your codes work in the 'Output' area. If your code is OK, you can save your project with your student ID."

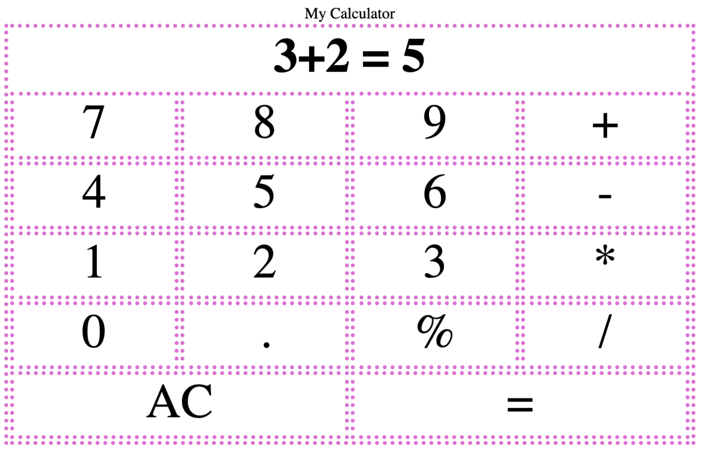


Figure 9: Sample *Calculator* page for assignment #2.

### 5.3.2 Hint

In this assignment, the hints are given by:
"For *Calculator*, you will need numbers, operators, and display area for the results.
For the *layout interface*, you can create your free design for your calculator. In the following example video, table layout and onclick function is used to get the inputs. You can also see Problem #25 for the reference.
For *JavaScript library functions*, you can use any arithmetic function for calculation in JavaScript. Here, simple *eval( )* function is used in the example. But, using *eval( )* in real-world applications is far more dangerous. We used it here for keeping our project simple."

### 5.3.3 Result and Discussion

The same seven students completed this project assignment using the table layout design. Their implementations of the arithmetic operations are similar but not the same among them as in the first project assignment.

## 6 Conclusion

This paper proposed the *code modification problem (CMP)* in *programming learning assistant system (PLAS)* for self-study of *web client programming using JavaScript*. The proposed CMP is designed for students to read a source code carefully by asking changes of parameters, values, or messages there, and to understand how to use the HTML/CSS elements and JavaScript library functions appearing in the code. Students can submit the answers until achieving the correct answers and *string matching* is used to check the correctness of any answer.

For the evaluation of the proposal, 25 CMP instances, which cover the basic functions and topics of *web client programming*, were generated, and 37 first-year master students in Okayama University, who have not taken a formal JavaScript programming course, were assigned to solve them. Based on the results of their solutions, the validity of the proposal has been confirmed. Additionally, two simple project assignments were assigned to the students to evaluate learning effects in code writing abilities.

In future works, we will create CMP instances for web server programming and for other commonly used concepts and functions in web client programming, and provide them to students to evaluate the effectiveness. We will also assist students to complete the project assignments by giving hints on layout designs and code implementations.

## References

[1] S.H. Jensen, A. Moller, P. Thiemann, "Type analysis for JavaScript," International Static Analysis Symposium, 238–255, 2009, doi:10.1007/978-3-642-03237-0_17.

[2] S.T. Aung, N. Funabiki, Y.W. Syaifudin, H.H.S. Kyaw, S.L. Aung, N.K. Dim, W.-C. Kao, "A proposal of grammar-concept understanding problem in Java programming learning assistant system," Journal of Advances in Information Technology, **12**(4), 342–350, 2021, doi:10.12720/jait.12.4.342-350.

[3] K. K. Zaw, N. Funabiki, W.-C. Kao, "A proposal of value trace problem for algorithm code reading in Java programming learning assistant system," Information Engineering Express, **1**(3), 9–18, 2015, doi:10.52731/iee.v1.i3.39.

[4] S. H. M. Shwe, N. Funabiki, Y. W. Syaifudin, E. E. Htet, H. H. S. Kyaw, P. P. Tar, N. W. Min, T. Myint, H. A. Thant, W.-C. Kao, "Value trace problems with assisting references for Python programming self-study," International Journal of Web Information Systems, **17**(4), 287–299, 2021, doi:10.1108/IJWIS-03-2021-0025.

[5] N. Funabiki, Tana, K.K. Zaw, N. Ishihara, W.-C. Kao, "A graph-based blank element selection algorithm for fill-in-blank problems in Java programming learning assistant system," IAENG International Journal of Computer Science, **44**(2), 247–260, 2017.

[6] H.H.S. Kyaw, N. Funabiki, S.L. Aung, N.K. Dim, W.-C. Kao, "A study of element fill-in-blank problems for C programming learning assistant system," International Journal of Information and Education Technology, **11**(6), 255–261, 2021, doi:10.18178/ijiet.2021.11.6.1520.

[7] H.H.S. Kyaw, S.S. Wint, N. Funabiki, W.-C. Kao, "A code completion problem in Java programming learning assistant system," IAENG International Journal of Computer Science, **47**(3), 350–359, 2020.

[8] N. Funabiki, Y. Matsushima, T. Nakanishi, N. Amano, "A Java programming learning assistant system using test-driven development method," IAENG International Journal of Computer Science, **40**(1), 38–46, 2013.

[9] I. Arawjo, C.-Y. Wang, A.C. Myers, E. Andersen, F. Guimbretière, "Teaching programming with gamified semantics," in 2017 CHI Conference on Human Factors in Computing, 4911–4923, 2017, doi:10.1145/3025453.3025711.

[10] J. Appleton, "Introducing intelligent exercises to support web application programming students," in 2017 International Conference on Information Communication Technologies in Education (ICICTE), 216–225, 2017.

[11] M. Uehara, "JavaScript development environment for programming education using smartphones," in 2019 International Symposium on Computing and Networking Workshops (CANDARW), 292–297, 2019, doi:10.1109/CANDARW.2019.00058.

[12] P. Vostinar, "Interactive course for JavaScript in LMS Moodle," in 2019 IEEE International Conference on Emerging eLearning Technologies and Applications (ICETA), 810–815, 2019, doi:10.1109/ICETA48886.2019.9039987.

[13] R. Maskeliūnas, A. Kulikajevas, T. Blažauskas, R. Damaševičius, J. Swacha, "An interactive serious mobile game for supporting the learning of programming in JavaScript in the context of eco-friendly city management," Computers, **9**(4), 1–18, 2020, doi:10.3390/computers9040102.

[14] I. S. Zinovieva, V. O. Artemchuk, A. V. Iatsyshyn, O. O. Popov, V. O. Kovach, A. V. Iatsyshyn, Y. O. Romanenko, O. V. Radchenko, "The use of online coding platforms as additional distance tools in programming education," Journal of Physics: Conference Series, **1840**, 2021, doi:10.1088/1742-6596/1840/1/012029.

[15] F. W.-B. Li, C.Watson, "Game-based concept visualization for learning programming," in 2011 International ACM Workshop on Multimedia Technologies for Distance Learning (MTDL), 37–42, 2011, doi:10.1145/2072598.2072607.

[16] E. Ünal, H. Çakir, "Students' views about the problem based collaborative learning environment supported by dynamic web technologies," Malaysian Online Journal of Educational Technology, **5**(2), 1–19, 2017.