

A Study of Grammar Concept Understanding and Value Trace Problems in JavaScript Programming Learning Assistant System

March, 2024

Khin Thet Mon

Graduate School of
Natural Science and Technology

(Doctor's Course)
OKAYAMA UNIVERSITY

Dissertation submitted to
Graduate School of Natural Science and Technology
of
Okayama University
for
partial fulfillment of the requirements
for the degree of
Doctor of Philosophy in Engineering

Written under the supervision of

Professor Nobuo Funabiki

and co-supervised by

Professor Satoshi Denno

and

Professor Yasuyuki Nogami

OKAYAMA UNIVERSITY, March 2024.

TO WHOM IT MAY CONCERN

We hereby certify that this is a typical copy of the original doctor thesis of
Ms. Khin Thet Mon

Signature of
the Supervisor

Seal of

Prof. Nobuo Funabiki

Graduate School of
Natural Science and Technology

Abstract

The ever-evolving landscape of web developments has witnessed a transformative paradigm shift, propelled by the ubiquity of *HTML*, *CSS*, and *JavaScript* in both desktop and mobile applications. *JavaScript*, in particular, has emerged as a linchpin, playing pivotal roles in browser detection, webpage customization, security enhancement, form validation, and the implementation of special effects, establishing itself as the foremost programming language. Distinguished by its accessible entry point, *JavaScript* allows learners to abstract away intricate programming details, focusing on fundamental concepts. Its unique integration with web browsers eliminates the need for elaborate development environments, making it an indispensable technology for crafting interactive and dynamic websites.

In response to the burgeoning interest in *JavaScript* for web client programming, this thesis addresses the discernible demand for high-quality learning tools, especially tailored for self-study at home. The contribution, *JavaScript Programming Learning Assistant System (JSPLAS)*, builds upon the success of *Java Programming Learning Assistant System (JPLAS)*, and is designed to assist learners across diverse proficiency levels of *JavaScript programming*.

JSPLAS encompasses a versatile array of exercise problems, each equipped with an automated marking system, streamlining the evaluation process and empowering students to engage in effective self-studies. This initiative aligns with the broader shift towards online, self-paced learning, providing students with the flexibility to acquire essential programming skills on their own terms.

This thesis delves into the development of *JSPLAS*, aiming to create an educational tool that aligns with the evolving landscape of web development. The proposed structured approach in *JSPLAS*, from foundational grammar concept studies to advanced code writing exercises, mirrors best practices in programming education, providing structured and incremental learning experiences.

As the first contribution of this thesis, I focus on the *Grammar-Concept Understanding Problem (GUP)* as an introductory step in the exploration of *web-client programming* using *JavaScript*. A *GUP* instance comprises a source code and a set of questions, each inquiring about the actual value of a crucial variable or an output message within the provided code. The assessment of correctness is achieved through *string matching* with predefined correct answers. *GUP* instances, integral to this study, comprise source codes and questions, with automated assessments ensuring prompt feedback to learners.

As the second contribution of this thesis, I present the *value trace problems (VTP)* for code reading study by novice students. I focus *VTP* as an integral component of the exploration into *JavaScript programming* education. A *VTP* instance comprises a source code and a set of questions, each inquiring about the actual value of a crucial variable or an output message within the provided code. The assessment of correctness is achieved through *string matching* with predefined correct answers. This study delves into the application of *VTP* for *code reading* in the context of *JavaScript programming*, shedding light on its efficacy as a pedagogical tool. I contribute to the ongoing efforts to enhance learning experiences in *JavaScript programming* by investigating and refining

methodologies for effective comprehension and skill development.

Based on these two contributions, as the third contribution of this thesis, I contribute the *Grammar-Concept Understanding Problem (GUP)* for learning the front-end programming using the *Vue JS* framework. This progressive *JavaScript* framework is approachable, performant, and versatile for building web user interfaces.

In subsequent phases of this research initiative, the generation of Grammar-Concept Understanding Problems (GUP) instances specific to JavaScript will be undertaken. These instances are designed to facilitate the in-depth study of advanced grammar concepts, prevalent libraries, and the intricacies of web application systems. Subsequently, these instances will be systematically allocated to students enrolled in JavaScript programming courses, thereby serving as a foundational tool for enhancing their comprehension and practical application of these concepts.

Simultaneously, an emphasis will be placed on refining references related to challenging topics. This improvement initiative aims to enhance the clarity and depth of resources available to students grappling with intricate subjects. The augmentation of references will be guided by a commitment to elucidating complex concepts and providing students with comprehensive support in navigating demanding areas of the curriculum.

Furthermore, Value Trace Problems (VTP) instances will be newly generated to cater to the study of advanced grammar concepts, common libraries, and data structures and algorithms within the JavaScript domain. These instances are designed to provide students with practical problem-solving scenarios, enabling a more robust understanding of the theoretical underpinnings discussed in the course.

These refined VTP instances, paralleling the theoretical curriculum, will be seamlessly integrated into the instructional framework of JavaScript programming courses. Their application will serve to challenge students in the practical application of acquired knowledge, thereby fostering a deeper understanding of the subject matter.

In summary, the upcoming research initiatives will encompass the generation and deployment of GUP instances for advanced grammar and web application systems, along with an enhancement of references for challenging topics. Additionally, new VTP instances will be devised to reinforce the study of advanced grammar, common libraries, and data structures and algorithms, further enriching the learning experience for students enrolled in JavaScript programming courses.

Acknowledgements

I am sincerely grateful to those who supported and encouraged me throughout my Ph.D. studies at Okayama University, Japan. Completing this thesis wouldn't have been possible without their invaluable assistance, and finding the right words to express my gratitude is a challenge. To all who have been part of this journey, you are the greatest blessing in my life.

My deepest appreciation goes to my supervisor, Professor Nobuo Funabiki, whose unwavering support and wisdom guided me through every phase of my thesis. I owe a great debt to him for his patience, advice, and encouragement, not only in scientific matters but also in life. Professor Funabiki provided invaluable insights, feedback, and guidance during the formulation of problems, code implementation, experimentation, paper writing, and presentations. Thanks to him, I stand as the person I am today.

I extend my heartfelt thanks to my co-supervisors, Professor Satoshi Denno and Professor Yasuyuki Nogami, for their continuous support, guidance, thoughtful suggestions, and proofreading of this work.

Special acknowledgment goes to Associate Professor Minoru Kuribayashi for his valuable insights during my research. I also express gratitude to the instructors who enriched my Ph.D. study with their profound knowledge.

My sincere thanks go to my friends and colleagues who supported me during this study, including Ms. Khaing Hsu Wai, Ms. Htoo Htoo Sandy Kyaw, Ms. Soe Thandar Aung, Ms. Ei Ei Htet, and all the members of the FUNABIKI Lab. Your support during challenging times and the shared thoughts and experiences mean a lot to me.

I am profoundly grateful to my husband, Myo Thant Swe, whose unwavering love, understanding, and encouragement sustained me throughout this arduous journey. His patience, support, and belief in my abilities provided me with the strength and determination to overcome challenges and persevere in pursuit of my academic goals. His sacrifices and unwavering commitment to our family have been my source of inspiration and motivation. I am truly blessed to have him by my side, and I dedicate this achievement to his unwavering love and support.

Lastly, my eternal gratitude goes to my beloved family and teachers, whose constant encouragement and support have been my strength and inspiration to overcome any difficulties in my life.

List of Publications

Journal Paper

1. **K. T. Mon**, K. H. Wai, N. Funabiki, and H. H. S. Kyaw, “Value trace problems for code reading study of JavaScript programming”, International Journal of Information and Education Technology (IJIET), vol. 12, no. 5, pp. 430-435, May 2022.
2. K. H. Wai, N. Funabiki, **K. T. Mon**, M. Z. Htun, S. H. M. Shwe, H. H. S. Kyaw, and W. C. Kao, “ A proposal of code modification problem for self-study of web client programming using JavaScript”, Advances in Science, Technology and Engineering Systems Journal (ASTESJ), vol. 7, no. 5, pp. 53-61, September 2022.

International Conference Paper

3. **K. T. Mon**, N. Funabiki, S. T. Aung, K. H. Wai and K. S. Lin, “ A study of grammar-concept understanding problems for web client programming with JavaScript”, International Symposium on Socially and Technically Symbiotic Systems (STSS), November 15-17, 2021. (Okayama Convention Center, Okayama, Japan)
4. **K. T. Mon**, H. Qi, K. H. Wai, N. Funabiki, W. C. Kao, “ A Study of Grammar-Concept Understanding Problems for Web-Client Programming”,
5. K. H. Wai, N. Funabiki, **K. T. Mon**, and S. H. M. Shwe, H. H. S. Kyaw, and K. S. Lin “ A proposal of code modification problem for Web client programming using JavaScript”, The Ninth International Symposium on Computing and Networking (CANDAR), pp. 196-202, November 23 - 26, 2021. (Online)
6. K. H. Wai, N. Funabiki, S. T. Aung, **K. T. Mon**, H. H. S. Kyaw, and W. C. Kao, “An implementation of answer code validation program for code writing problem in Java programming learning assistant system”, 2023 11th International Conference on Information and Education Technology (ICIET 2023), pp. 193-198, March 18-20, 2023. (Shonan Institute of Technology, Fujisawa, Japan)

Other Papers

7. H. Qui, N. Funabiki, K. H. Wai, M. Z. Htun, **K. T. Mon**, and W. C. Kao, “ A study of element fill-in-blank problems for applicative grammar topics in JavaScript-based web-client programming” 情理学会研究告, コンピュータと教育研究会 vol. 2022-CE-166, no. 2, October 2022. (10月1日海大学湘南キャンパス)

List of Figures

2.1	Server platform for JPLAS.	6
2.2	Software architecutre for JPLAS.	7
4.1	VTP Instance User Interface in JSPLAS.	27

List of Tables

3.1	Generated GUP Instances Statistics.	19
3.2	Student Results for GUP Instances	21
4.1	Performance Metrics for VTP-I	29
4.2	Performance Metrics for VTP-II	30
4.3	Performance Metrics for VTP-I	32
4.4	Performance Metrics for VTP-I	33

List of Codes

3.1	JavaScript Code	20
4.1	JavaScript Code	26
5.1	VUE JS Code	35
5.2	VUE JS Code	36

Contents

Abstract	i
Acknowledgements	iii
List of Publications	iv
List of Figures	vi
List of Tables	vii
List of Codes	viii
1 Introduction	1
1.1 Background	1
1.2 Contributions	3
1.3 Contents of This Dissertation	5
2 Review of JPLAS	6
2.1 Overview of JPLAS	6
2.1.1 Server Platform	6
2.1.2 Software Architecture	6
2.1.3 Problem Types	7
2.2 Service Functions in JPLAS	9
2.2.1 Teacher Service Functions	9
2.2.2 Student Service Functions	9
2.2.3 Extended HTML Source Code Download Functions for JSPLAS	10
2.3 Desktop-version JPLAS	10
2.4 Offline Answering Functions in JPLAS	11
2.4.1 Operational Workflow	11
2.4.2 File Generation	12
2.4.3 Cheating Prevention	12
2.5 Extension of JPLAS to JSPLAS	12
2.5.1 JavaScript's Integral Role in Modern Web Development	12
2.5.2 Attractions to Study JavaScript	12
2.5.3 Elaboration of JSPLAS to JPLAS	13
2.5.4 Problem Types of JSPLAS	13
2.6 Summary	14

3	Grammer-Concept Understanding Problem	15
3.1	Introduction of GUP	15
3.2	Overview of GUP	15
3.3	JSPLAS GUP Extension	16
3.4	Design of GUP Instances	17
3.5	GUP Instances Generation	17
3.5.1	Design Goals of GUP	17
3.5.2	Generation Procedures	18
3.6	Generated Problems	18
3.6.1	Generated GUP Instances	18
3.6.2	Example GUP Instances	20
3.6.3	User Interfaces	20
3.7	Evaluations in JSPLAS-GUP	21
3.7.1	Analysis of Solution Results	21
3.7.2	Analysis of Submission Times	21
3.8	Benefits of JSPLAS GUPs	22
3.9	Summary	23
4	Value Trace Problems	24
4.1	Introduction to VTP	24
4.2	Methodology	24
4.3	JSPLAS and VTP Integration	24
4.3.1	VTP Instance Generation Procedure	25
4.3.1.1	Source Code Selection	25
4.3.1.2	Identification of Important Messages and Key Variables	25
4.3.1.3	Addition of Standard Output Statements	25
4.3.1.4	Question Preparation and Correct Answer Assignment	25
4.3.1.5	Compilation into Text File	25
4.3.1.6	Program Execution for Answer Interface	26
4.3.1.7	Registration of VTP Instance for Assignment	26
4.4	Example of VTP Instance: JavaScript Data Types Usage	26
4.5	Evaluation	27
4.5.1	Student Background	28
4.5.2	Evaluation Process	28
4.5.3	Results and Observations	28
4.5.4	Student Performance Analysis in VTP-I and VTP-II	30
4.5.5	Overall Implications	33
4.6	Summary	33
5	Grammer-Concept Understanding Problems : VUE Js	34
5.1	Introduction	34
5.2	Importance of studying Vue JS Framework	34
5.3	Extension of JSPLAS GUP problems with Vue JS Problems	35
5.3.1	Example of VTP Instance: JavaScript Data Types Usage	35
5.4	Summary	37
6	Related Works	38

7 Conclusion	39
References	40

Chapter 1

Introduction

1.1 Background

In today's web development landscape, HTML, CSS, and JavaScript form the essential trio for creating dynamic, interactive web applications. JavaScript, in particular, takes center stage, serving crucial functions such as browser detection, webpage customization, security reinforcement, form validation, and the orchestration of captivating special effects. As the primary programming language, JavaScript defines the modern web experience.

Yet, as technology evolves, it's essential to not only recognize JavaScript's current dominance but also stay attuned to shifting trends. New languages and frameworks may emerge, potentially altering the programming landscape. This exploration aims to delve into web development's multifaceted terrain, examining JavaScript's central role, current challenges, and the dynamic forces shaping the digital language landscape.

JavaScript's prominence in modern software developments stems from distinct features that set it apart:

- **Ease of Entry:**
JavaScript abstracts complexities, making it beginner-friendly and allowing a focus on core programming concepts.
- **In-Browser Development:**
Unique to JavaScript is the ability to code and test directly in web browsers, streamlining the development process without extensive setup.
- **Web Technology Backbone:**
Teaming up with HTML and CSS, JavaScript is foundational in creating dynamic, and interactive websites.
- **Rich Prewritten Functionalities:**
A robust set of predefined functionalities accelerates development, enabling programmers to build complex systems efficiently. In exploring JavaScript's landscape, these features collectively underscore its pivotal role in shaping modern software development.

JavaScript serves as a compelling catalyst for novice programmers, seamlessly translating raw data into interactive and captivating visualization. Its user-friendly nature makes it an ideal can-

didate for introduction in the early years of STEM (Science, Technology, Engineering, and Mathematics) education, offering students a hands-on experience with fundamental programming concepts.

Given these educational dynamics, there's a growing interest among individuals to delve into JavaScript for web client programming. The demand for high-quality learning tools has surged, particularly for self-study at home, as many students lack the opportunity to enroll in formal courses at schools. Presently, JavaScript is a staple in many university web programming courses, underscoring its significance in shaping the skillsets of aspiring programmers.

This research underscores a dedicated commitment to cultivating impactful, self-directed learning experiences in JavaScript programming. With a deliberate focus on meeting the increasing demand for accessible and comprehensive learning resources, this study aims to make a significant contribution to the education of individuals aspiring to master the intricacies of JavaScript programming. Through the exploration and development of effective learning tools, this research strives to empower learners in their journey towards proficiency in JavaScript, aligning with the evolving landscape of web development education.

In the forthcoming stages of this research, the exploration of JavaScript programming will persist, with a particular emphasis on expanding the breadth of topics covered. The ongoing development of the *JavaScript Programming Learning Assistant System (JSPLAS)* will witness an augmentation of exercise types to encompass a more comprehensive array of challenges.

A pivotal focus will be directed toward the integration of real-world projects within the learning curriculum. This initiative aims to provide learners with immersive hands-on experiences, exposing them to practical scenarios commonly encountered in professional JavaScript development. The incorporation of real-world projects is anticipated to bridge the gap between theoretical understanding and practical application, fostering a more robust and industry-aligned learning experience within JSPLAS.

By delving into these future endeavors, the overarching goal remains steadfast: to advance the capabilities of JSPLAS as a dynamic and effective platform for self-directed learning in JavaScript programming. Through the exploration of diverse topics and the incorporation of real-world projects, this thesis seeks to contribute to the evolution of JavaScript education, ensuring that learners are equipped with both a broad understanding of key concepts and practical skills demanded in professional settings.

Up to this point, I have crafted the *Java Programming Learning Assistant System (JPLAS)* to facilitate self-study of Java programming.

JPLAS encompasses a diverse range of exercise problems designed to enhance Java programming skills. These include:

- Grammar-Concept Understanding Problem (GUP): Focusing on understanding Java grammar and concepts.
- Value Trace Problem (VTP): Involving the tracing of variable values in code execution.
- Element Fill-in-Blank Problem (EFP): Testing knowledge through filling in blanks in code elements.
- Code Amendment Problem (CAP): Evaluating the ability to make specific amendments to a given code.
- Code Modifying Problem (CMP): Assessing proficiency in modifying existing code.

- Code Writing Problem (CWP): Encouraging the creation of original code solutions.

In the commitment to providing self-study environments for novice students interested in JavaScript programming, I have initiated the development of the *JavaScript Programming Learning Assistant System (JSPLAS)*. This project builds upon and extends the foundational work laid out in *JSLAS*, tailoring the learning experience to the distinctive features and nuances of JavaScript. The aim is to empower students in their independent exploration of JavaScript, fostering a comprehensive and interactive learning environment.

To ensure the accessibility of *JSPLAS* in diverse settings, I have implemented both a web-based online version and a desktop-based offline version. The online *JSPLAS* utilizes *Ubuntu* as the operating system, *Tomcat* as the web application server, *JSP* for application programs with HTML, and MySQL for the database handling problem descriptions and student data. Users can conveniently access *JSPLAS* through a standard web browser.

Recognizing that the online version may not be optimal for students in areas with limited internet access, I have developed the Desktop version *JSPLAS*. This offline version operates independently of online databases and web servers. In Desktop Version *JSPLAS*, assignment files and answer text files can be easily shared between teachers and students through USB memories, file servers, or email, ensuring flexibility and adaptability to various learning environments.

Effective programming learning often begins with students solving simple problems, focusing initially on code reading and grammar concepts studies. As they progress, the emphasis shifts to practicing coding problems, particularly those involving code modification. Consequently, in *JSPLAS*, students are encouraged to follow this logical sequence of problem types.

JSPLAS has several types of exercise problems to cover a variety of students at different learning levels, such as *Grammar-Concept Understanding Problems (GUP)* and *value trace problem (VTP)*. In JavaScript programming studies using *JSPLAS*, a student is expected to solve the exercises along this order of problem types.

1.2 Contributions

As the first contribution, the focus of this thesis is on the *Grammar-Concept Understanding Problem (GUP)*, serving as the foundational step in advancing proficiency in web client programming with JavaScript. Each GUP instance comprises a source code, a set of questions, and their corresponding correct answer strings. These questions prompt students to discern the definition of pre-determined terms in fundamental grammar concepts, such as methods, tags, or attributes, within the provided source code.

The assessment of student responses in the *Grammar-Concept Understanding Problem (GUP)* and the subsequent automated generation of new GUP instances for web client programming rely on a string matching mechanism with the correct string. This meticulous process ensures precision in evaluating students' comprehension of predefined terms within the source code.

To streamline the generation of new GUP instances, I have adapted a generation program that was initially tailored for Java. This modification involves the integration of a specialized keyword list and a set of questions designed with meticulous attention to the nuanced intricacies of web client programming. This tailored approach aims to present students with challenges that are not only relevant but also aligned with the distinctive demands of JavaScript in the context of web development.

In the evaluations, I created 25 GUP instances, incorporating 209 questions sourced from basic web client programming found in websites and textbooks. These instances were then assigned to

26 master's students at Okayama University who had not previously studied JavaScript in academic settings. The results revealed a remarkable success rate, with all students correctly solving all questions, except for two instances where a 99% accuracy was achieved. This outcome strongly supports the effectiveness of the approach in facilitating the understanding of fundamental concepts in web client programming with JavaScript among novice students.

The *Value Trace Problem (VTP)* within JSPLAS is meticulously crafted to facilitate the study of fundamental grammar concepts and programming skills through code reading, especially tailored for novice students. Each VTP instance comprises a source code, a set of questions, and their respective correct answer strings. These questions prompt students to determine the value of significant variables or output messages within the provided source code. The accuracy of student responses is gauged through string matching with the predetermined correct string.

In this study, I delve into the *Value Trace Problem (VTP)* in JSPLAS, building upon the groundwork laid in JPLAS. To explore basic grammar concepts and JavaScript programming implementations, I curate a dataset of 57 source codes encompassing fundamental grammar concepts. Subsequently, I meticulously analyze crucial variables, integrate their standard outputs into the codes, and manually prepare the correct answers. Following this, I generate corresponding *VTP* instances using a dedicated program designed for creating the answer interface.

In the evaluation phase, I distribute the 57 generated *VTP* instances among students in both Myanmar and Japan who are currently engaged in learning JavaScript programming. Through this process, I seek to affirm the efficacy of the approach in facilitating the understanding of basic JavaScript programming concepts.

To enhance the learning experience, I categorize the instances into two groups: 32 in *VTP-I* and 25 in *VTP-II*. This division aims to assist students in solving problems in a step-by-step manner, providing a structured and progressive approach to mastering JavaScript programming fundamentals.

The design goals of the Value Trace Problem (*VTP*) are outlined as follows:

1. **Varied Source Codes for Effective Programming Study:** Present a diverse set of source codes that are impactful for programming study. Depict these codes in their full forms to cater to the understanding of novice students.
2. **Emphasis on Code Reading and Understanding:** Ensure that students can accurately answer the questions solely through careful reading and understanding of the provided source codes.
3. **Automated Evaluation through String Matching:** Facilitate an automated marking process where any student's response can be assessed through string matching, streamlining the evaluation procedure.

The third contribution of the thesis is the extension of the JSPLAS for one of the most popular *JavaScript Framework Vue.js*. *Vue.js* has garnered significant popularity within the front-end development community. Recognized as a progressive JavaScript framework, *Vue.js* excels in constructing user interfaces, boasting features that contribute to its widespread adoption. Its appeal lies in its simplicity, flexibility, and gradual learning curve, making it a preferred choice for developers engaged in projects of diverse sizes and complexities. *Vue.js*'s strengths include reactive data binding and component-based architecture, facilitating the seamless creation of dynamic and interactive user interfaces. Staying informed about the latest developments is advisable, given that the landscape of technologies can evolve.

I offer GUP problems for *Vue.js* that can be easily understood in the basics. It is recommended that students initially focus on solving straightforward problems that involve code reading and

understanding grammar concepts. Following this foundational step, they can progress to more advanced practice by working on coding problems that involve modifying existing code. This sequential approach ensures a gradual and comprehensive development of programming skills, starting with fundamental comprehension and progressing to practical applications through code modifications.

1.3 Contents of This Dissertation

The remaining part of this thesis is organized as follows: Chapter 2 reviews the *JavaScript Programming Learning Assistant System (JSPLAS)*, where I discuss the problem types, the user service functions, and evaluations. Chapter 3 presents the *Grammar-Concept Understanding Problems (GUP)* in JSPLAS. Chapter 4 presents the *Value Trace Problem (VTP)* in JSPLAS. Chapter 5 presents the *Grammar-Concept Understanding Problems (GUP)* for Vue JS in JSPLAS. Chapter 6 presents previous works related to this thesis. Finally, Chapter 7 concludes this thesis with some future works.

Chapter 2

Review of JPLAS

In this chapter, I review the *Java Programming Learning Assistant System (JPLAS)* and introduce the *JavaScript Programming Learning Assistant System (JSPLAS)*.

2.1 Overview of JPLAS

First, I review the system architecture, implemented problem types, and user service functions in *JPLAS*.

2.1.1 Server Platform

JPLAS has been implemented as a web application system. In the server architecture depicted in Figure 2.1, *Node.js* is employed as the web application server, complemented by the *Express.js* framework. In realizing the MVC model-based application, Java is utilized for the model (M), executing JUnit for testing answer source codes in code-writing problems. The view (V) is constructed using EJS, CSS, JavaScript, while JavaScript is employed for the controller (C).

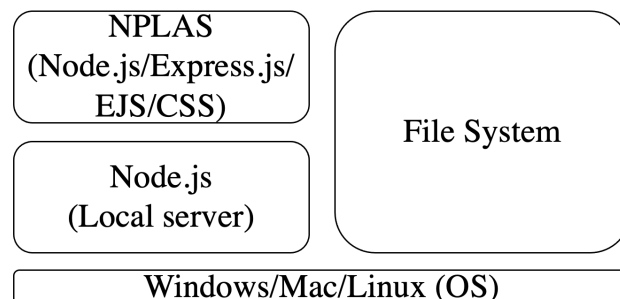


Figure 2.1: Server platform for JPLAS.

2.1.2 Software Architecture

The software architecture of *JPLAS (Java Programming Learning Assessment System)*, which is illustrated in Figure 2.2, adheres to the *Model-View-Controller (MVC)* paradigm, a prevalent architectural pattern in web application systems.

For the *model (M)* part of MVC model, *JUnit* is used where *Java* is used to implement the programs. The *file system* is used to manage the data where every data is provided by a file. In terms of the *view (V)* part, the platform uses *Embedded JavaScript (EJS)* as the template engine, rather than the usual template engine of *Express.js* (a web application framework for Node.js). For the *control (C)* part, *Node.js* and *Express.js* are adopted as the server-side technologies, where *JavaScript* is used to implement the programs.

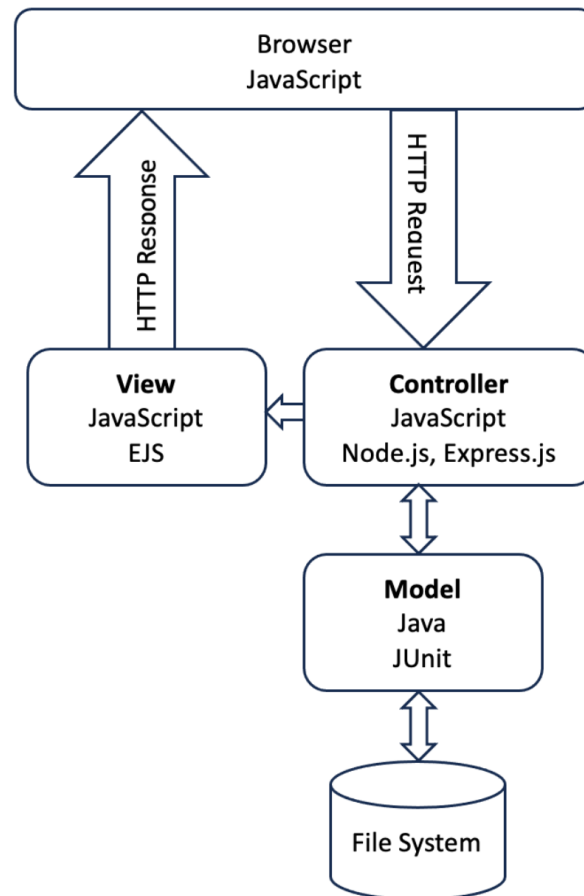


Figure 2.2: Software architecture for JPLAS.

2.1.3 Problem Types

JPLAS (Java Programming Learning Assessment System) incorporates a diverse array of exercise problems, catering to the varied proficiency levels of students. The distinct problem types within JPLAS are delineated as follows:

1. Grammar Concept Understanding Problem (GUP):
 - Comprises a source code and inquiries about grammar concepts or code behaviors.
 - Answers may take the form of numbers, words, or short sentences, evaluated through string matching.

- Features an algorithm that autonomously generates GUP instances by extracting registered keywords, associating questions with each keyword, and identifying required data in the correct answer from the code.

2. Value Trace Problem (VTP):

- Challenges students to trace the actual values of crucial variables during code execution.
- Answer correctness is assessed by comparing student responses with stored correct values on the server.

3. Element Fill-in-blank Problem (EFP):

- Mandates students to complete blank elements within a given Java code.
- Correctness is determined by comparing student inputs with the original code's stored elements, considered unique correct answers for the blanks.
- Features a proposed blank element selection algorithm to assist teachers in generating feasible problems.

4. Statement Fill-in-blank Problem (SFB):

- Tasks students with filling in blank statements in a code.
- Code correctness is gauged using JUnit, an open-source software for the *test-driven development (TDD)* method.
- Utilizes the *program dependency graph (PDG)* to aid teachers in selecting blank statements from a code.

5. Code Writing Problem (CWP):

- Requires students to compose an entire code from scratch, adhering to specifications outlined in the test code.
- Code correctness is evaluated against the provided test code.

6. Code Amendment Problem (CAP):

- Presents students with a Java source code containing missing or erroneous elements (problem code).
- Students identify and rectify the locations of these elements, with correctness determined through string matching against the original code.

7. Code Completion Problem (CCP):

- Displays a source code with unspecified missing elements to students.
- Students locate and fill in these missing elements.
- Correctness is verified through string matching for each statement in the answer against the corresponding original statement in the code.
- Any single incorrect element renders the entire answer incorrect.

These diverse problem types collectively contribute to the richness of JPLAS's assessment capabilities, providing a comprehensive platform for evaluating students' Java programming skills across various dimensions.

2.2 Service Functions in JPLAS

In JPLAS, two distinct categories of service functions are delineated: *teacher service functions* and *student service functions*. These functions cater to the unique needs and roles of educators and learners within the system. Additionally, an expansion has been made in the realm of HTML source code download functions, augmenting the capabilities of JSPLAS.

2.2.1 Teacher Service Functions

Within the framework of JPLAS, teacher service functions play a pivotal role, offering a suite of features meticulously designed to enhance pedagogical efficacy and streamline the assessment process. These functions encompass the dynamic generation of programming problems, seamless registration and management of assignments, the creation of comprehensive projects consolidating acquired concepts, and a nuanced analysis of student performance. The system provides educators with the capability to evaluate assignment difficulty based on student submission patterns, enabling informed adjustments or replacements to maintain an optimal challenge level. Further, the incorporation of hint and recommendation functions empowers teachers to guide students through challenges, fostering improved understanding. The creation of topic references and the ability to offer targeted instruction to individual students exhibiting exceptional performance underscores the system's commitment to personalized and effective teaching strategies. Additionally, JPLAS facilitates the creation of project assignments that synthesize previously learned concepts, offering a holistic assessment of students' grasp of overarching topics within the curriculum. These teacher service functions collectively contribute to a robust instructional environment, aligning with the system's objective of providing a comprehensive and adaptive learning experience in the realm of Java programming.

2.2.2 Student Service Functions

The student service functions within JPLAS encompass essential features for efficient engagement, including the viewing of assignments, solving project assignments, and submitting answers. Particularly, in the case of code writing problem types, students are tasked with crafting source code by interpreting the problem statement and adhering to the specifications outlined in the accompanying test code. The process of handling student answers in JPLAS involves several sequential steps:

1. Upon accessing JPLAS, students are presented with a list of assigned problems.
2. Upon selecting a specific problem, the corresponding problem text from the database is displayed.
3. Students then proceed to write their answers within the designated forms.
4. Submitted answers are marked on the server, with both the answers and marking results being stored in the database.
5. JPLAS provides feedback to the students based on the evaluation.
6. If necessary, students have the option to iterate through the process, revisiting and modifying their answers.

The procedural workflow for both teacher and student interactions with JPLAS is outlined as follows:

1. A teacher initiates the generation of a new problem and registers it in the database.
2. The teacher curates a new assignment by selecting pertinent problems from the database and registers it accordingly.
3. A student selects an assignment to solve.
4. Within the chosen assignment, a student selects a specific problem to address.
5. Students proceed to solve the questions posed in the problem, submitting their answers to the server.
6. The server undertakes the task of marking the submitted answers and returns the corresponding marking results.
7. In the event of inaccuracies, students have the flexibility to modify incorrect answers and resubmit them to the server.
8. Students can review their solution results for assignments.
9. Teachers, in turn, have access to the solution results of all students for the assignments, enabling a comprehensive evaluation of student performance.

This structured utilization procedure underscores the bidirectional functionality of JPLAS, facilitating both educators and students in the seamless execution of their respective roles within the platform.

2.2.3 Extended HTML Source Code Download Functions for JSPLAS

This extension pertains to the enhancement of HTML source code download capabilities within JSPLAS. It implies an augmentation of functionalities related to retrieving and downloading HTML source codes. This enhancement could serve various purposes, such as enabling users to obtain code snippets, templates, or completed exercises for reference or offline use.

2.3 Desktop-version JPLAS

As previously discussed, the inherent challenges posed by unstable Internet access, particularly in developing countries with weak network infrastructure and frequent power shortages, prompted the development of an alternative solution to enhance accessibility. In response, an offline desktop version of *JPLAS* called *D-JPLAS* has been implemented to cater to schools and homes facing constraints in Internet connectivity. *D-JPLAS* operates exclusively on the client's personal computer without requiring server access through the Internet. Notably, it circumvents the challenges associated with unstable connectivity and the potential absence of skilled personnel for web server management. Unlike the online version, *D-JPLAS* functions autonomously on the user's PC, housing all programs and data, including problems and student answers, within the local file system, eliminating the need for a centralized database.

In the *D-JPLAS* environment, students submit their answers to teachers using USB memories, facilitating a manual transfer of information. Subsequently, teachers analyze student responses on their own PCs and provide feedback directly to the students. This offline version aims to ensure a seamless educational experience even in areas with limited or unreliable internet access. Within the scope of this thesis, a dedicated function for analyzing student answers in *D-JPLAS* has been meticulously implemented, further enhancing its utility as a robust and accessible solution for educational institutions and homes facing challenges related to internet connectivity.

2.4 Offline Answering Functions in JPLAS

As highlighted in Section 2.3, the implementation of an offline answering function within JPLAS serves as a crucial enhancement, allowing students to respond to problems even in situations where direct access to the JPLAS server is hindered due to the unavailability of the Internet. This feature proves to be not only beneficial but indeed essential in the practical application of JPLAS. The offline JPLAS enables students to engage with problem instances, facilitating problem assignment delivery and answer submission through the use of a USB device. The offline functionality is structured around three primary functions: operation flow, file generation, and cheating prevention. These functions collectively contribute to the seamless operation and integrity of the offline JPLAS, ensuring an effective and secure educational experience for students even in scenarios where online connectivity is limited or unavailable.

2.4.1 Operational Workflow

The operational workflow of the offline answering function unfolds through a systematic series of steps:

1. Problem Instance Download:
 - A teacher accesses the JPLAS server online, selects specific problem instances for the assignment, and downloads the requisite files onto their personal computer.
2. Assignment Distribution:
 - The teacher disseminates the assignment files to students either via a file server or USB memories, enabling offline access to the assigned problems.
3. Assignment Answering:
 - Students receive and install the downloaded files on their individual computers, proceeding to answer the problem instances offline.
 - Using web browsers offline, the correctness of each answer is promptly verified through the utilization of a JavaScript program embedded in the browsers.
4. Answering Result Submission:
 - Upon completion, students submit their final answering results to the teacher using either a file server or USB memories, ensuring a secure and offline transfer of information.
5. Answering Result Upload:
 - Subsequently, the teacher, upon receiving students' answering results, uploads the compiled data back to the JPLAS server online for centralized management and assessment.

This meticulously designed offline operation flow not only circumvents challenges associated with internet connectivity but also ensures the seamless integration of offline and online functionalities. The use of USB memories or file servers facilitates the secure exchange of information between teachers and students, allowing for effective assessment and management of students' performance within the *JPLAS* framework even in environments with limited or no internet access.

2.4.2 File Generation

Table 2.1 delineates the essential files, along with their specifications, for the offline answering functionality of *JPLAS*. These files are specifically tailored for the problem view, the answer marking, and the answer storage.

2.4.3 Cheating Prevention

In the offline iteration of *JPLAS*, the imperative task is to disseminate correct answers to students for prompt verification on their respective browsers. To safeguard against the inadvertent exposure of correct answers, a precautionary measure involves generating hash values using SHA256. Furthermore, to mitigate the likelihood of identical hash values for identical correct answers, the assignment ID and the problem ID are concatenated with each correct answer before undergoing the hashing process. This concatenation strategy ensures distinct hash values for the same correct answers associated with different blanks, thereby preserving independence among blanks.

2.5 Extension of JPLAS to JSPLAS

2.5.1 JavaScript's Integral Role in Modern Web Development

In contemporary web application systems for both desktop and mobile devices, the ubiquitous use of web client programming, incorporating HTML, CSS, and JavaScript, has become the norm. JavaScript, among these three languages, assumes pivotal roles in various aspects of web development. Its functionalities extend to detecting browser types, customizing webpages, bolstering security measures, validating forms, and incorporating captivating special effects into the overall webpage experience. Consequently, JavaScript stands out as the preeminent and most widely adopted programming language in the context of modern web development.

2.5.2 Attractions to Study JavaScript

JavaScript, owing to its versatile applications, has sparked a substantial demand for developers across the spectrum, ranging from fledgling startups to industry behemoths. Mastery of this language not only opens avenues for remote freelancing opportunities but also positions individuals for lucrative full-time roles, making it a pivotal skill set in the dynamic landscape of modern web development careers.

Here are some key reasons why studying JavaScript is valuable.

1. **Ubiquitous Web Language:** JavaScript is an integral part of web development, allowing developers to create interactive and dynamic websites. It is supported by all major web browsers, making it a universal language for client-side scripting.

2. **Versatility and Flexibility:** JavaScript is a versatile language that can be used for both front-end and back-end development. Its flexibility enables developers to build a wide range of applications, from simple website enhancements to complex server-side applications.
3. **High Demand in Job Market:** The widespread use of JavaScript has led to a high demand for skilled JavaScript developers. Learning JavaScript enhances employability and opens up opportunities in various industries, from startups to established tech companies.
4. **Community Support and Resources:** JavaScript has a vibrant and active developer community. This means ample resources, libraries, and frameworks are available, making it easier for developers to find support, share knowledge, and stay updated on best practices.
5. **Compatibility with Frameworks and Libraries:** JavaScript seamlessly integrates with popular frameworks and libraries such as React, Angular, and Vue.js, enhancing its capabilities and making it easier to build scalable and efficient applications.
6. **Cross-Platform Development:** With the advent of technologies like *Node.js*, JavaScript can be used for server-side development, enabling developers to create applications that run on both the client and server. This fosters a more unified and streamlined development process.
7. **Continuous Evolution:** JavaScript is continually evolving, with regular updates and new features. Staying current with JavaScript allows developers to leverage the latest advancements, ensuring their skills remain relevant in the ever-changing tech landscape.
8. **Client-Side Interactivity:** JavaScript enables the creation of dynamic and interactive user interfaces, enhancing the overall user experience. This is crucial for modern web applications that prioritize user engagement and responsiveness.

2.5.3 Elaboration of JSPLAS to JPLAS

In light of the aforementioned scenarios, an increasing number of individuals are expressing keen interest in learning JavaScript for web client programming. This surge in demand has underscored the need for high-quality learning tools, particularly for students engaged in self-study at home. Many learners lack the opportunity to enroll in formal courses at educational institutions. Presently, numerous universities are incorporating JavaScript into their web programming courses, recognizing its significance in contemporary development practices.

Recognizing the paramount importance of JavaScript, I have decided to expand the *Java Programming Learning Assistant System (JPLAS)* to include JavaScript programming. This extension, aptly named the *JavaScript Programming Learning Assistant System (JSPLAS)*, aims to provide comprehensive support and resources for individuals seeking to enhance their skills in JavaScript development.

At present, I have successfully integrated a diverse array of problems equipped with automatic marking functions. This inclusive approach is designed to cater to self-studies in JavaScript programming, accommodating learners at different proficiency levels, particularly novice students.

2.5.4 Problem Types of JSPLAS

I have successfully incorporated a variety of problem types into JSPLAS, ensuring a comprehensive learning experience. These problem types cater to diverse aspects of JavaScript programming, providing valuable resources for learners at different proficiency levels.

Here are the problem types that I already implemented in JSPLAS

1. Grammar-concept Understanding Problem (GUP)
2. Value Trace Problem (VTP)
3. Code Modification Problem (CMP)

Within this thesis, particular emphasis is placed on the GUP and VTP. Across a majority of the problem types, student responses are assessed through string matching against correct codes. In select instances, especially in specific problem types, the evaluation involves unit testing through designated test codes.

In continuation, I have endeavored to expand our curriculum by introducing an in-depth section dedicated to one of the most popular JavaScript frameworks, *Vue.js*. This addition aims to provide learners with comprehensive insights and practical knowledge, further enriching their understanding of modern JavaScript development practices.”

2.6 Summary

This chapter provides an extensive review of *Java Programming Learning Assistant System (JPLAS)*, encompassing its various functions and desktop versions. Additionally, the chapter introduces and elucidates *JPSLAS* as an extension of *JPLAS*, expanding the scope and capabilities of the learning assistant system.

Chapter 3

Grammar-Concept Understanding Problem

In this chapter, I present the *JavaScript Programming Learning Assistant System (JSPLAS)* and introduce the *Grammar-Concept Understanding Problems (GUP)* in JSPLAS.

3.1 Introduction of GUP

For effective programming skill development, students should begin the study by dissecting simple codes for fundamental concepts. Subsequently, they should focus on grammar and syntax to deepen understanding. Practices through code modification exercises to reinforce learning will be effective. Finally, they should engage in more complex programming exercises for practical application and skill refinements. This structured approach lays a solid foundation for writing proficient and well-structured code.

The *GUP (Grammar-Concept Understanding Problem)* serves as the initial step in studying web client programming with JavaScript. It is recommended that students start their exploration of web client programming with JavaScript by initially concentrating on solving uncomplicated problems, such as those involving code reading and understanding grammar concepts. After establishing this foundational understanding, students can then advance their skills by engaging in more complex coding problems that require modifications of existing source codes. This step-by-step learning approach can guarantee a systematic and thorough development of programming skills, beginning with fundamental comprehension and advancing progressively to practical applications through code modifications.

3.2 Overview of GUP

A GUP instance comprises a source code, a series of questions, and their corresponding correct answer strings. Each question delineates the definition of a predetermined term related to a basic grammar concept, representing a method, tag, or attribute found in the provided source code. The question prompts the responder to identify and answer the corresponding term within the code.

The accuracy of a student's response is evaluated through string matching with the correct string. To facilitate the automatic creation of a new GUP instance, the generation program coded in Java has been developed for web client programming. This program needs to define a set of keywords and questions, which are described by the keyword list and the question list for effective web client programming studies.

3.3 JSPLAS GUP Extension

In the landscape of web developments, JavaScript stands out as a pivotal language, playing indispensable roles in web application systems for both desktop and mobile devices. This prominence can be attributed to its unique combination of accessibility, immediate feedback through browser-based coding, essential integration with HTML and CSS, and the availability of pre-written functionalities. Despite facing challenges, such as browser compatibility issues, JavaScript remains the most popular programming language, facilitating the creation of interactive and dynamic web experiences.

JavaScript's capacity to seamlessly translate raw data into interactive visualizations positions it as a potent motivator for novice programmers, making it an ideal introduction to basic programming concepts in early STEM education. As a result, the growing interest in learning JavaScript for web client programming has spurred a demand for high-quality learning tools, particularly for self-study at home, where access to formal courses may be limited. Despite its integral role in web programming courses at universities, the increasing popularity of JavaScript as an educational tool suggests a need for versatile and accessible resources for learners of all backgrounds.

JavaScript's ability to effortlessly translate data into engaging visualizations makes it a compelling entry point for novice programmers, particularly in early STEM education. The rising interest in learning JavaScript for web client programming, coupled with the demand for accessible self-study tools, highlights the need for high-quality resources. Despite its integration into university courses, the popularity of JavaScript suggests a broader, diverse audience seeking versatile learning options.

- Motivator for Novice Programmers

JavaScript's knack for turning data into interactive visuals motivates beginners, sparking interest in programming concepts.

- Early Introduction in STEM Education

Early integration of JavaScript in STEM education introduces fundamental programming concepts and ignites interest in practical applications.

- Rising interest in learning JavaScript

The growing interest in learning JavaScript for web programming underscores its pivotal role in creating interactive online experiences.

- Demand for High-Quality Learning Tools

The shift towards self-study emphasizes the need for accessible, high-quality learning tools that cater to diverse learning preferences.

- Integration in University Courses

While JavaScript is part of many university courses, the demand for self-directed learning highlights the need for adaptable resources beyond traditional academic settings.

The upswing in individuals pursuing JavaScript for web client programming highlights the need for accessible learning tools, particularly for those engaged in self-study due to limited formal opportunities. As universities integrate JavaScript into their courses, the demand for versatile

resources becomes more pronounced, emphasizing the importance of catering to diverse learning environments and preferences.

As the demand for *JavaScript learning tools* continues to rise, our decision to extend our existing learning platform *JPLAS* to *JSPLAS* is a proactive response to the changing landscape of programming education. By tailoring our existing system to address the specific needs of JavaScript learners, I aim to contribute to the accessibility and effectiveness of programming education, particularly in the context of self-directed learning.

3.4 Design of GUP Instances

This section introduces the design principles underlying *GUP* instances, providing a comprehensive overview of the key features and considerations that shape the structure of our instances.

The *Grammar-Concept Understanding Problem (GUP)* in *JSPLAS* caters to novice students by offering targeted questions related to grammar concepts and the relationship between JavaScript and HTML. Each GUP problem focuses on predefined question words, encompassing methods, tags, attributes, events, and others. The assessment of answer correctness relies on string matching with predefined correct answers, ensuring an objective evaluation of students' understanding. Building upon the success of GUP in *JPLAS*, this approach tailors the learning experience for novices, emphasizing the integration of JavaScript codes within HTML pages as they embark on their initial stages of programming education.

A GUP instance in *JSPLAS* comprises a source code, a set of questions, and their corresponding correct answer strings. These questions elucidate the definitions of predefined words, embodying basic grammar concepts like methods, tags, or attributes present in the given source code, prompting students to identify and match the correct instances in the code. The correctness of students' responses is assessed through string matching with the predefined correct strings. The adaptation of the GUP instance generation program for web client programming facilitates the automatic generation of new GUP instances, incorporating updated keyword and question lists tailored to the nuances of web client programming, thus enhancing the versatility of the learning tool.

3.5 GUP Instances Generation

In this section, I introduce the GUP generation program, specifically crafted to support teachers in crafting new instances within the domain of JavaScript learning. This algorithm functions as a valuable tool, empowering educators to generate customized GUP instances that cater to the distinct needs and preferences of users engaged in the study of JavaScript programming.

3.5.1 Design Goals of GUP

This section outlines the core design goals of *Grammar-Concept Understanding Problems (GUP)*, driving its development and evaluation to enhance programming studies.

1. Automatic GUP Instance Generation: GUP aims to automatically generate instances by identifying pertinent source codes, enhancing adaptability for diverse programming scenarios.
2. Diverse Source Codes for Learning: GUP provides a variety of source codes and explanations, catering to students at different proficiency levels and offering comprehensive learning experiences.

3. **Conceptual Definitions Memorization:** GUP supports the memorization of programming conceptual definitions by presenting source codes in a way that encourages careful comprehension.
4. **Automatic Conceptual String Matching for Assessment:** Automation is the key in GUP, utilizing conceptual string-matching algorithms for efficient and accurate assessments of student answers, and providing timely feedback.

In summary, these design goals underpin GUP's developments, focusing on automation and comprehensive learning experiences in programming education.

3.5.2 Generation Procedures

This section delineates the systematic process employed for the generation of GUP instances, a critical aspect of the proposed educational platform.

1. **Modification of GUP Generator:** To initiate the generation of a new GUP instance, the GUP generator undergoes modification. This entails adapting the program to create a text file that encapsulates the instance's questions and correct answers. This file is dynamically updated by modifying both the keyword list and the question list.
2. **Selection of Suitable Source Code:** The process involves the careful selection of a source code that encapsulates fundamental grammar concepts. The chosen code must include pivotal keywords, forming the foundation for effective learning.
3. **Execution of GUP Generator Subsequently,** the modified GUP generator is executed, incorporating the selected source code. This action generates a comprehensive text file, serving as the blueprint for the GUP instance.
4. **Utilization of Interface Generator:** Another program, the answer interface generator, is then invoked using the generated text file. This secondary program is responsible for translating the GUP instance into HTML/CSS/JavaScript files, which collectively form the interactive answer interface accessible through a web browser.
5. **Assignment of GUP Instance to Students:** The final step involves the allocation of the newly generated GUP instance to students. This ensures that learners have access to a customized and dynamically created learning resource.

3.6 Generated Problems

First, I present the generated GUP instances, example GUPs, and user interfaces.

3.6.1 Generated GUP Instances

Table 3.1 shows the grammar concept, the number of lines (LOC) in the source code, the number of questions, and the average correct answer rate of the students for the 25 GUP instances.

Table 3.1: Generated GUP Instances Statistics.

ID	Grammar Concept	LOC	# of Questions	Avg Correct Rate
1	Output by using innerHTML	11	5	100%
2	Output by using innerHTML	13	10	99%
3	How to change the label of the button	14	8	100%
4	Get the ID of the clicked button	21	7	100%
5	Get the HTML elements by Query selector	33	12	100%
6	How to use the alert message	14	6	100%
7	How to use event handlers	14	3	100%
8	How to change the HTML attribute values	10	7	100%
9	Event handler when the user leaves from the input field	27	10	100%
10	How to use on-submit function to redirect other page	19	11	99%
11	How to use the select event handler	17	8	100%
12	How to use the reset event handler	17	5	100%
13	How to use the key-up and key-down event handler	17	8	100%
14	How to use the confirm window object	20	7	100%
15	How to use the prompt window object	15	7	100%
16	How to use the open window object	14	5	100%
17	How to use the open window object (timeout setting)	17	7	100%
18	Validate form	25	10	100%
19	JavaScript retype password validation	27	10	100%
20	Email validation	26	12	100%
21	How to show date	15	10	100%
22	Digital clock	37	13	100%
23	Eval function for calculation	52	14	100%
24	Timer	35	9	100%
25	Error handling	26	5	100%
Average		21.44	8.36	100%
Total (SD)		9.82	2.81	0%

3.6.2 Example GUP Instances

The following source code is used in ID=1. It uses `innerHTML`, and contains five keywords, `script`, `document`, `getElementById`, `innerHTML`, and `id`, which must be understood by the students.

Source Code

```
1 <html>
2 <head>
3 </head>
4 <body>
5 <h1>Output by using innerHTML</h1>
6 <p id="msg"></p>
7 <script>
8 document.getElementById("msg").innerHTML="HelloWorld";
9 </script>
10 </body>
11 </html>
```

Listing 3.1: JavaScript Code

The following five questions with the correct answers are selected for the source code by the GUP generator:

1. What is the main tag to define the JavaScript part in this program? (script)
2. What is the pre-defined object that represents the whole HTML document in this program? (document)
3. What is the pre-defined method to return the html element that has the value for the id attribute in the argument in this program? (getElementById)
4. What is the pre-defined property to set or return the HTML message or contents of an element in this program? (innerHTML)
5. What is the attribute to specify a unique id for an HTML element in this program to access and manipulate the element with the specific id? (id)

3.6.3 User Interfaces

The answer interface (UI) designed for this educational coding instance serves as a crucial element in facilitating effective learning. Upon completion of the answer forms, students trigger the evaluation process by clicking the “Answer” button. The correct response maintains the white background, while the incorrect one prompts a visual cue with the red background. This immediate feedback mechanism encourages iterative learning as students can submit answers repeatedly. Additionally, the accessibility feature allowing students to download the source code provides a practical dimension to their learning, fostering hands-on experimentation and deeper understanding. The UI’s strategic use of color-coded feedback and the iterative learning environment collectively contribute to a dynamic and engaging educational experience.

3.7 Evaluations in JSPLAS-GUP

In this section, I evaluate the 25 GUP instances within the context of web client programming, involving 26 first-year master students at Okayama University. It is pertinent to note that, before the application of these instances, there was no dedicated class on web client programming. This evaluation aims to assess the effectiveness of the GUP instances as an educational tool for students with varying levels of programming experience, offering insights into their impacts on the learning outcomes in the absence of a formalized instructional framework.

3.7.1 Analysis of Solution Results

Upon a closer examination of the solution results detailed in Table 3.2, it is noteworthy that a significant majority of the first-year master students at Okayama University exhibited exemplary performances. Specifically, 23 out of the 26 students achieved the perfect of 100% correctness rate, underscoring the effectiveness of the GUP instances. Additionally, one student demonstrated the commendable 98% accuracy, while another achieved the 99% success rate. This outcome suggests that the designed instances are well-calibrated, providing an appropriate level of challenge for novice students embarking on the study of web client programming. The results affirm the suitability of the educational approach in fostering a high level of comprehension and proficiency among the participating students.

3.7.2 Analysis of Submission Times

The examination of submission times provides valuable insights into the interactive learning process facilitated by JSPLAS. Notably, students were granted the flexibility to repeatedly submit their answers for marking. Table 2???? (where?) illustrates the distribution of students across different ranges of submission times. Remarkably, one student demonstrated exceptional engagement by submitting answers only 26 times, achieving the 100% correct rate across the 25 instances. Conversely, an interesting observation arises as three students engaged in nearly 400 submissions each. This equates to 16 times the number of instances and twice the number of questions. In practical terms, these students submitted answers 16 times for a single instance and twice for each question on average. This distinctive pattern of extensive submissions underscores diverse approaches to mastering the material, providing valuable insights into individual learning strategies within the framework of JSPLAS.

Table 3.2: Student Results for GUP Instances

Group	# of Submissions	Size
I	26	1
II	27-50	6
III	51-100	6
IV	101-200	7
V	201-300	3
VI	301-400	3

3.8 Benefits of JSPLAS GUPs

Here are some benefits of the JSPLAS GUPs.

- **Focused Grammar Instruction:**
The system provides targeted grammar instruction within the context of JavaScript programming, helping learners understand and apply grammar rules specific to the language.
- **Integrated Learning Environment:**
The inclusion of Grammar-Concept Understanding Problems within a JavaScript learning system creates a cohesive and integrated environment for learners, ensuring that they grasp both programming concepts and proper language usage simultaneously.
- **Practical Application of Grammar:**
Grammar-Concept Understanding Problems likely involve practical scenarios related to JavaScript programming, allowing learners to apply grammar rules in real-world coding situations.
- **Enhanced Code Readability:**
Understanding the grammar specific to JavaScript contributes to writing cleaner and more readable code. This skill is crucial for effective collaboration in programming teams and maintaining codebases.
- **Immediate Feedback for Code Quality:**
The system provides immediate feedback on the grammatical correctness of the code, helping learners identify and rectify syntax errors promptly. This aids in the development of good coding practices.
- **Adaptive Learning Paths:**
The learning system may adapt its content based on individual student performance in grammar-related exercises, tailoring the learning path to address specific areas of difficulty.
- **Varied Engagement Levels:**
The inclusion of Grammar-Concept Understanding Problems accommodates learners with different levels of grammatical proficiency, allowing them to engage at a pace that suits their skill levels.
- **Comprehensive Skill Development:**
Learners gain not only JavaScript programming skills but also proficiency in expressing these skills with proper grammar. This comprehensive approach contributes to well-rounded expertise.
- **Increased Code Quality:**
A focus on grammar improves the overall quality of the code by promoting adherence to best practices and coding conventions. This is essential for writing maintainable and scalable JavaScript applications.
- **Preparation for Professional Development:**
Exposure to grammar-specific problems in JavaScript programming prepares learners for professional development, where adherence to coding standards is crucial for success in the industry.

- Individualized Learning Experience:
Through adaptive learning paths and varied engagement levels, the system tailors the learning experience to individual learners, ensuring that each student can progress at his/her own pace.

3.9 Summary

In summary, this chapter delved into the examination of the *Grammar-Concept Understanding Problem (GUP)* within the realm of web client programming, specifically focusing on novice students. The study involved the creation of 25 GUP instances, encompassing 209 questions, formulated with source codes incorporating essential grammar concepts in HTML, CSS, and JavaScript. The application of these instances to 26 students at Okayama University yielded highly favorable results, as each participant attained the correct rate of 98% or higher across all instances. This substantiates the effectiveness of GUP instances as a beneficial tool for reinforcing foundational grammar understanding in the context of web client programming. Looking ahead, future works will extend the scope to generate GUP instances catering to advanced grammar concepts, common libraries, and comprehensive web application systems. These instances will be integrated into JavaScript programming courses, offering a platform for ongoing exploration and evaluation of their educational efficacy.

Chapter 4

Value Trace Problems

In this chapter, I present the *Value Trace Problem (VTP)* in *JavaScript Programming Learning Assistant System (JPLAS)*.

4.1 Introduction to VTP

This section introduces the *Value Trace Problem (VTP)* methodology employed for self-study in JavaScript programming. The purpose of VTP is to prompt learners to trace the actual values of critical variables or output messages within the provided source code. Subsequently, correctness is assessed through string matching against predetermined correct responses.

In this contribution, I delve into the exploration of the *Value Trace Problem (VTP)* within the context of code reading studies in *JavaScript Programming Learning Assistant System (JSPLAS)* by novice students. Recognizing the increasing significance of JavaScript in the development of web applications, this study focuses on assessing the educational efficacy of VTP instances. I aim to bridge the gap in JavaScript education, particularly in university curricula where the subject might be underrepresented.

4.2 Methodology

To facilitate code reading studies, I have generated 57 *VTP* instances leveraging diverse source codes that encapsulate fundamental JavaScript grammar concepts. These instances serve as targeted exercises for novice learners, emphasizing comprehension of essential programming structures.

Our approach involves a meticulous analysis of the identified source codes, with a specific emphasis on discerning important variables. Standard outputs for these variables are incorporated into the codes, and correct answers are manually prepared. Subsequently, I leverage a program designed for generating the answer interface to create corresponding VTP instances.

4.3 JSPLAS and VTP Integration

JavaScript's prominence in web application development underscores its importance in modern programming. Despite its relevance, many universities still do not offer comprehensive JavaScript education. The inherent difficulty in learning JavaScript is exacerbated by idiosyncratic coding

styles. In response, I am actively developing the *JavaScript Programming Learning Assistant System (JSPLAS)* to support self-study initiatives, building upon the success of the *Java Programming Learning Assistant System (JPLAS)*.

JSPLAS extends our previous works in *JPLAS*, offering a tailored learning experience for JavaScript programming. The cornerstone of our approach is the incorporation of the VTP within code reading studies. Each VTP instance in *JSPLAS* comprises a source code snippet accompanied by a set of questions. These questions prompt learners to identify the values of critical variables or output messages within the code. The correctness of student responses is evaluated through string matching against pre-registered correct answers.

The generation process involves systematically applying the answer interface program to the analyzed source codes, producing 57 distinct VTP instances. This method ensures the alignment of the generated instances with the identified important variables and their associated standard outputs.

4.3.1 VTP Instance Generation Procedure

The generation of a *Value Trace Problem (VTP)* instance involves a systematic and structured procedure aimed at creating educational challenges for learners. The following steps outline the method for generating a VTP instance:

4.3.1.1 Source Code Selection

I begin by selecting source codes that are apt for studying basic grammar concepts or fundamental data structures/algorithms in the context of JavaScript programming.

4.3.1.2 Identification of Important Messages and Key Variables

Then, I analyze the selected source code to identify crucial messages and key variables. These elements are fundamental for crafting meaningful questions that promote a comprehensive understanding of the code.

4.3.1.3 Addition of Standard Output Statements

I also augment the source code by incorporating standard output statements corresponding to the identified important messages and key variables. This step ensures that the generated VTP instance aligns with the intended learning objectives.

4.3.1.4 Question Preparation and Correct Answer Assignment

I formulate the questions that prompt the examination of messages or values at the standard output. Simultaneously, I manually prepare correct answers to serve as benchmarks for evaluating learner responses.

4.3.1.5 Compilation into Text File

I consolidate the modified source code, questions, and correct answers into a single text file. This file serves as the input for the subsequent steps in the generation process.

4.3.1.6 Program Execution for Answer Interface

I utilize the dedicated programs capable of processing the text file and generating HTML/CSS/-JavaScript files. These files collectively form the answer interface, accessible through a web browser.

4.3.1.7 Registration of VTP Instance for Assignment

Once the answer interface files are generated, I register the VTP instance within the assignment framework for distribution to students. This ensures that the educational content is seamlessly integrated into the learning platform.

By adhering to this comprehensive procedure, educators and developers can systematically generate VTP instances, fostering an interactive and engaging learning experience for students studying JavaScript programming.

4.4 Example of VTP Instance: JavaScript Data Types Usage

In this section, we provide an illustrative example of a *Value Trace Problem (VTP)* instance generated as part of this study. The example pertains to the study of JavaScript Data Types Usage and specifically focuses on instance ID=1 within the basic grammar of *VTP*.

Source Code

```
1 main( ) {  
2   var undeclaredVar;  
3   var obj = null;  
4   var num = 100;  
5   var inProgress = true;  
6   var Greeting = "Hello";  
7   console.log(undeclaredVar); // undefined  
8   console.log(obj); // null  
9   console.log(num); // number  
10  console.log(inProgress); // boolean  
11  console.log(Greeting); // string  
12 }
```

Listing 4.1: JavaScript Code

Questions

1. What is the value of undeclaredVar? _1_
 2. What is the value of obj? _2_
 3. What is the value of num (approximately two decimal places)? _3_
 4. What is the value of inProgress? _4_
 5. What is the value of Greeting? _5_
-

```
/* Demonstration of Data types Usage */

main( )
{
    var undeclaredVar;
    var obj = null;
    var num = 100;
    var inProgress = true;
    var Greeting = "Hello";

    console.log(undeclaredVar);    // undefined
    console.log(obj);              // null
    console.log(num);              // number
    console.log(inProgress);       // boolean
    console.log(Greeting);         // string
}

Results

What is the value of undeclaredVar ? 
What is the value of obj ? 
What is the value of num(approximately two decimal place)? 
What is the value of inProgress? 
What is the value of Greeting? 
```

Answer

Answer

Top

Figure 4.1: VTP Instance User Interface in JSPLAS.

Figure 4.1 provides an overview of the user interface designed for the Value Trace Problem (VTP) instance. The interface is structured to facilitate student interaction with the following features:

- **Answer Form Completion:** Students are required to fill out all answer forms corresponding to the questions provided.
- **”Answer” Button:** Upon completion, students must click the blue “Answer” button to submit their responses for evaluation.
- **Feedback Mechanism:** In the event of an incorrect answer, the background color of the form will dynamically change to red. Conversely, a correct answer results in a white background.
- **Iterative Submission:** Students have the flexibility to submit answers repeatedly until all responses are correct, fostering an interactive and adaptive learning experience.

This intuitive user interface aims to enhance the learning process by providing real-time feedback, encouraging students to refine their understanding and mastery of JavaScript programming concepts through continuous engagement.

4.5 Evaluation

This section shows a comprehensive evaluation of the 57 generated Value Trace Problem (VTP) instances by applying them to a diverse group of 45 university students in Myanmar and Japan.

The student demographics include 11 third-year undergraduate students in Myanmar, who possess prior experience in JavaScript programming in a web programming course. Additionally, 34 first-year master students in Japan engaged with JavaScript programming for the first time, preparing for the evaluation through provided reference materials.

4.5.1 Student Background

- Myanmar Cohort: Eleven third-year undergraduate students with prior exposure to JavaScript programming.
- Japan Cohort: Thirty-four first-year master's students were introduced to JavaScript programming through provided references.

4.5.2 Evaluation Process

The evaluation process involved distributing the 57 VTP instances to the student cohorts, prompting them to analyze the code, respond to questions, and articulate their understanding of traced values and messages.

4.5.3 Results and Observations

The majority of students actively participated in the evaluation, showcasing notable engagement in both cohorts. However, it is important to note that one student did not provide answers for VTP-I instances.

- Myanmar Cohort: Demonstrated a nuanced understanding and proficiency in handling VTP instances, leveraging their prior exposure to JavaScript programming.
- Japan Cohort: Despite being introduced to JavaScript programming through reference materials, first-year master's students exhibited commendable efforts, highlighting the effectiveness of the educational approach.

As previously discussed, the student participants were systematically divided into two distinct groups: 32 individuals for VTP-I and 25 individuals for VTP-II. The experimental design details, crucial for understanding the learning conditions in each group, are presented in Tables 4.1 and 4.2.

Table 4.1: Performance Metrics for VTP-I

ID	Basic Grammar Concept	LOC	# of forms	# of avg of correct rate
1	Data Types	13	5	98%
2	Arithmetic Operators	32	11	48%
3	Assignment Operators	48	7	86%
4	Functions	27	6	95%
5	Objects	22	3	98%
6	Strings	15	5	98%
7	BackSlash Characters	25	3	82%
8	String Specifications	28	4	89%
9	String Extraction	39	9	52%
10	String Replace and Trim	34	6	77%
11	String Conversion	41	9	98%
12	Number Methods	42	15	100%
13	Number Methods	29	8	100%
14	Number to String	40	10	57%
15	Number to String	28	8	98%
16	Arrays	55	11	84%
17	Arrays Association	29	4	91%
18	Basic Arrays Methods	36	8	89%
19	Basic Arrays Methods	29	6	89%
20	Arrays Changing	41	7	89%
21	Arrays Splicing	50	9	98%
22	Arrays Splicing	48	12	98%
23	Arrays Merging	39	8	91%
24	Arrays Sorting	46	8	93%
25	Conditions (if)	11	1	100%
26	Conditions (if... else)	13	2	100%
27	Conditions (if... elseif... else)	15	3	100%
28	Conditions (Switch)	58	3	98%
29	Looping (For)	13	11	100%
30	Looping (Do While)	9	5	100%
31	Looping (While)	9	5	100%
32	Looping (While)	9	3	100%
Average		30.4	6.7	90.27%
Total		973	215	(SD)13.87

In VTP-I, 39 out of the 44 students demonstrated a commendable level of proficiency, solving more than 90% of the 215 answer forms across 32 instances. Among the remaining five students, four achieved an 88% success rate, while one student exhibited the smallest rate, completing 80% of the forms.

Table 4.2: Performance Metrics for VTP-II

ID	Basic Grammar Concept	LOC	# of forms	# of avg of correct rate
1	Data Types (I)	33	4	100%
2	Data Types (II)	16	2	95%
3	Comparison Operators	51	12	95%
4	Logical Operators	38	3	95%
5	Conditional Operators	11	1	95%
6	Precedence Associativity	20	4	100%
7	Upper Case Check	24	2	70%
8	Vowel Checking	23	1	100%
9	Prime Number Check	34	3	98%
10	Data Types Check	30	3	91%
11	Perfect Number Check	28	2	91%
12	Global Functions (Eval())	15	2	100%
13	Global Functions (isFinite())	17	2	95%
14	Global Functions (isNaN())	37	12	93%
15	Global Functions (parseFloat())	20	7	93%
16	Global Functions (parseInt())	31	12	95%
17	Global Functions (Number())	24	9	91%
18	Numeric Array Sorting	46	3	95%
19	Dates	33	5	98%
20	Var VS Let – Block Scope	18	6	98%
21	Var VS Let – Redeclaration	17	2	98%
22	Var VS Let – Looping	16	2	98%
23	Variable Declaration (const)	25	5	93%
24	Var VS Const – Block Scope	11	3	98%
25	Arrow Functions	18	3	95%
Average		25.44	4.36	94.91%
Total		636	110	(SD)5.73

In VTP-II, 43 out of the 45 students demonstrated a high level of proficiency, correctly solving more than 90% of the 110 answer forms across 25 instances. Among the remaining two students, one achieved an 88% success rate, while the other displayed the smallest rate, solving 83% of the forms.

4.5.4 Student Performance Analysis in VTP-I and VTP-II

In this section, I present a detailed analysis of student performance metrics for the VTP-I and VTP-II. Tables 4.3 and Tables 4.4 provide a comprehensive overview, including student ID, the number of correctly solved answer forms, total answer submissions, incomplete instances, and the correct answer rate for each student.

Student ID	# of correct forms	# of answer submissions	# of incomplete instances	Correct answer rate
1	215	32	0	100%
2	215	32	0	100%

Continued on next page

Table 4.3 – continued from previous page

Student ID	# of correct forms	# of answer submissions	# of incomplete instances	Correct answer rate
3	215	32	0	100%
4	215	34	0	100%
5	215	34	0	100%
6	215	43	0	100%
7	214	108	0	99%
8	213	35	0	99%
9	213	53	0	99%
10	212	38	0	99%
11	212	43	0	99%
12	212	48	0	99%
13	212	49	0	99%
14	211	68	1	98%
15	211	78	0	98%
16	210	33	0	98%
17	210	45	0	98%
18	210	50	0	98%
19	210	51	0	98%
20	210	52	0	98%
21	210	53	0	98%
22	210	86	0	98%
23	210	100	1	98%
24	210	205	0	98%
25	209	35	0	97%
26	209	36	0	97%
27	209	70	0	97%
28	207	34	1	96%
29	206	58	0	96%
30	205	65	0	95%
31	204	49	0	95%
32	200	42	1	93%
33	198	108	0	92%
34	198	115	0	92%
35	197	55	0	92%
36	195	49	0	92%
37	195	95	0	92%
38	195	110	0	92%
39	193	55	0	92%
40	190	34	0	88%
41	190	34	3	88%
42	190	81	1	88%
43	189	29	3	88%
44	171	36	21	81%
Total	9040	2585	137	(SD)4.51%
Continued on next page				

Table 4.3 – continued from previous page

Student ID	# of correct forms	# of answer submissions	# of incomplete instances	Correct answer rate
Average	205.45	58.75	3.11	96.55%

Table 4.3: Performance Metrics for VTP-I

In VTP-I, 39 out of the 44 students demonstrated a commendable level of proficiency, solving more than 90% of the 215 answer forms across 32 instances. Among the remaining five students, four achieved an 88% success rate, while one student exhibited the smallest rate, completing 80% of the forms.

Student ID	# of correct forms	# of answer submissions	# of incomplete instances	Correct answer rate
1	215	32	0	100%
2	215	32	0	100%
3	215	32	0	100%
4	215	34	0	100%
5	215	34	0	100%
6	215	43	0	100%
7	214	108	0	99%
8	213	35	0	99%
9	213	53	0	99%
10	212	38	0	99%
11	110	34	0	100%
12	110	39	0	100%
13	110	39	0	100%
14	110	40	0	100%
15	110	41	0	100%
16	110	45	0	100%
17	110	45	0	100%
18	110	45	0	100%
19	110	47	0	100%
20	110	70	0	100%
21	110	146	0	100%
22	109	25	0	99%
23	109	25	0	99%
24	109	25	0	99%
25	109	26	0	99%
26	109	27	0	99%
27	109	27	0	99%
28	109	28	0	99%
29	109	31	0	99%
30	109	36	0	99%
31	109	40	0	99%
32	109	58	0	99%
33	109	84	0	99%
34	108	27	1	98%

Continued on next page

Table 4.4 – continued from previous page

Student ID	# of correct forms	# of answer submissions	# of incomplete instances	Correct answer rate
35	108	31	0	98%
36	107	31	0	97%
37	107	47	0	97%
38	107	57	0	97%
39	107	81	0	97%
40	105	25	0	95%
41	105	32	8	95%
42	103	27	1	94%
43	103	65	0	94%
44	97	22	3	88%
45	91	26	0	83%
Total	9040	2585	137	(SD)4.51%
Average	205.45	58.75	3.11	96.55%

Table 4.4: Performance Metrics for VTP-I

In VTP-II, 43 out of the 45 students demonstrated a high level of proficiency, correctly solving more than 90% of the 110 answer forms across the 25 instances. Among the remaining two students, one achieved an 88% success rate, while the other displayed the smallest rate, solving 83% of the forms.

4.5.5 Overall Implications

The data presented in Tables III and IV highlight that the generated VTP instances in this paper are well-structured and aligned with proper difficulty levels. The majority of students excelled, particularly in surpassing the 90% correct answer rate, indicating that these instances are suitable for students initiating their studies in JavaScript programming. This outcome underscores the effectiveness and appropriateness of the designed VTP instances.

4.6 Summary

This paper presents a comprehensive study of the Value Trace Problem for code reading studies in JSPLAS by novice students. Through the generation and evaluation of the 57 VTP instances, I aim to contribute to the enhancement of JavaScript education, particularly in contexts where it remains underrepresented. Our ongoing development of JSPLAS signifies a commitment to providing a robust learning assistant system for JavaScript programming, leveraging the success and methodologies established in our previous work with JPLAS.

Chapter 5

Grammar-Concept Understanding Problems : VUE Js

In this chapter, I introduce the *Grammar-Concept Understanding Problems (GUP)* for Vue JS in JSPLAS.

5.1 Introduction

In the dynamic realm of front-end web development, *Vue.js* has emerged as a pivotal player, offering an open-source JavaScript framework tailored for the creation of user interfaces. Introduced in 2014 by Evan You, *Vue.js* has garnered increasing recognition within the developer community. Its appeal lies in its adeptness at constructing front-end applications that boast scalability, maintainability, and resource efficiency.

This framework operates at the forefront of modern web development practices, providing developers with a robust toolset for crafting responsive and dynamic user interfaces. *Vue.js's gradual ascent in popularity* is attributed to its inherent effectiveness, aligning seamlessly with the demands of contemporary web applications. The chapters will delve into the foundational principles, key features, and practical applications of *Vue.js*, unraveling the factors that have contributed to its acclaim and widespread adoption among developers.

5.2 Importance of studying Vue JS Framework

In this section, I present the importance of studying *Vue JS*.

1. Relevance to Industry Dynamics:

Vue.js has rapidly ascended in prominence, becoming an integral component of contemporary web development practices. Its widespread adoption by companies and development teams underscores its relevance to industry trends. By delving into the intricacies of *Vue.js*, students cultivate skills that directly address the demands of the current professional landscape.

2. Pedagogical Advantages and Gradual Learning Curve:

Vue.js is renowned for its incremental adaptability, rendering it an ideal framework for educational settings. Its approachable learning curve facilitates a systematic understanding of

fundamental front-end development concepts. This deliberate simplicity allows students to focus on core principles without unnecessary complexities.

3. Modular Architecture and Scalability:

The component-based architecture of Vue.js is conducive to modularity, enabling students to grasp the nuances of scalable application development. By navigating projects of varying magnitudes, students gain insights into the architectural considerations required for smaller and large-scale applications.

4. Introduction to Real-Time Data Binding:

Vue.js introduces students to the concept of reactivity, a paradigm wherein alterations in data are seamlessly propagated to the user interface in real time. This real-time data binding mechanism not only enhances the efficiency of web applications but also serves as a foundational aspect for students to comprehend in the context of developing responsive and interactive user interfaces.

5. Community Engagement and Abundance of Resources:

Vue.js boasts a dynamic and supportive community, providing an invaluable resource for students. The availability of comprehensive online tutorials, forums, and collaborative platforms fosters a culture of shared learning. This community-driven ethos equips students with the ability to seek guidance, share insights, and actively contribute to the broader Vue.js ecosystem.

6. Professional Readiness and Employability:

As students embark on their journey toward professional development, familiarity with Vue.js positions them advantageously. Many enterprises, ranging from established corporations to emerging startups, integrate Vue.js into their technological frameworks. Students proficient in Vue.js are thus well-prepared to make meaningful contributions from the outset of their professional careers.

5.3 Extension of JSPLAS GUP problems with Vue JS Problems

5.3.1 Example of VTP Instance: JavaScript Data Types Usage

In this section, I provide an illustrative example of a *Grammar-Concept Understanding Problems (GUP)* instance generated as part of this study. The example pertains to the study of Vue Framework Usage and specifically focuses on instance ID=1 within the basic grammar of *GUP*.

Source Code

```
1 #app.vue
2 <script setup>
3 import ButtonCounter from './ButtonCounter.vue'
4 </script>
5
6 <template>
7   <h1>Here are many child components!</h1>
8   <ButtonCounter />
9   <ButtonCounter />
10  <ButtonCounter />
11 </template>
```

Listing 5.1: VUE JS Code

```
1 #ButtonCounter.vue
2 <script setup>
3 import { ref } from 'vue'
4
5 const count = ref(0)
6 </script>
7
8 <template>
9   <button @click="count++">
10     You clicked me {{ count }} times.
11   </button>
12 </template>
```

Listing 5.2: VUE JS Code

Questions

1. Which syntax is an HTML-based template syntax that allows you to declaratively bind the rendered DOM to the underlying component instance's data? _1_
 2. Which keyword is used to import modules or values from other files in JavaScript? _2_
 3. Which is the name you give to the imported module or component? This name will be referred to the component in the current file. _3_
 4. Which tag is a section in a Vue single-file component where you specify the HTML structure that you want the component to render.? _4_
-

The above source code provides an overview of the sample VUE problems designed for the Grammar-Concept Understanding (GUP) instance.

5.4 Summary

In this chapter, I presented the *Importances for Studying Vue JS* with the extension of *Grammar-Concept Understanding Problems* in JSPLAS. I designed 10 problems at the basic understanding level. I will assign these exercises to the students who are willing to study Front-end web development technologies. I will evaluate the effectiveness of the approach by asking some students to study Vue.js by using *Vue.js GUP Problems*.

In future works, I will improve the GUP Problems by adding the advanced problems of *Vue.js*.

Chapter 6

Related Works

In this section, we introduce some related works to this thesis.

In [58], Gómez-Albarrán reviewed several tools to support the teaching and learning of programming and categorized them into four groups, as tools including simply reduced development environments, example-based environments, tools based on visualization and animation, and simulation environments.

In [59], Brusilovsky et al. presented QuizPACK which asks the value of a particular variable or a string with some fragment of a program. They demonstrate that it significantly improved the knowledge of semantics. Their approach is similar to our work in this paper. We will compare this study.

In [60], Kordaki et al. presented a computer-based problem-solving environment named LECGO (Learning Environment for programming using C using Geometrical Objects) for learning C programming by beginners. It emphasized on multiple external representations in student learning, motivation through performing problem-solving activities from the familiar and meaningful context, the active participation of students by using hands-on experience, appropriate feedback to aid self-corrections, and holistic, activity-based, multi-media, multi-representational and multi-layered content for learning basic concepts of C programming.

In [61], Lee et al. presented Gidget, a game where the eponymous robot protagonist is cast as a fallible character that blames itself for not being able to correctly write code to complete its missions. Players learn programming by debugging its problematic code.

In [62], Li et al. presented a game-based learning environment to support novice students learning programming. It exploits game construction tasks to make the elementary programming more intuitive to learn, and comprises concept visualization techniques, to allow students to learn key concepts in programming through game object manipulation.

In [63], Hwang et al. proposed the Web-based programming-assisted system for cooperation called WPASC, designing learning activities for facilitating cooperative programming learning, and investigated the cooperative programming learning behavior of students and its relationship with learning performance.

Chapter 7

Conclusion

In conclusion, this thesis has undertaken a comprehensive exploration into the challenges and learning dynamics inherent in JavaScript programming, with a specific focus on Grammar-Concept Understanding Problems (GUP) and the value trace problem (VTP) among novice students in *Java Programming Learning Assistant System (JPLAS)*. The investigation also extended to the realm of GUP Problems customized for the Vue JS Framework.

Our journey began by scrutinizing the *Grammar-Concept Understanding Problem (GUP)* in web client programming, tailoring 25 instances with 209 questions using HTML, CSS, and JavaScript. The subsequent application of these instances to 26 university students at Okayama University revealed their remarkable effectiveness, as every student achieved a 98% or higher correct rate for each instance. This underscores the pedagogical value of GUPs in fortifying fundamental grammar comprehension among novice learners.

Turning our attention to the *value trace problem (VTP)*, we generated 57 instances using diverse JavaScript source codes. The application of these instances to 45 university students in Myanmar and Japan yielded consistently positive results, with each student achieving an 80% or higher correct rate on average. These findings emphasize the global relevance of addressing conceptual challenges in JavaScript programming education and the effectiveness of tailored solutions.

In the final phase, we highlighted the importance of studying *Vue JS* and introduced an extension of Grammar-Concept Understanding Problems in JSPLAS. Designing ten problems at the basic understanding level aimed to enrich the learning experience for students interested in front-end web development technologies, particularly Vue.js. The proposed evaluation involves assigning these exercises to students, with the effectiveness of the approach measured through their study of Vue.js using the designated Vue.js GUP Problems.

In essence, this research contributes valuable insights to the field of programming education, offering tangible solutions to challenges faced by novice learners. The positive outcomes observed in the application of GUPs and VTP instances underscore the importance of tailored educational approaches. As technology and programming paradigms continue to evolve, understanding and addressing these challenges become pivotal for fostering a proficient and adaptive generation of programmers. This thesis serves as a stepping stone for future research and endeavors aimed at refining programming education methodologies and ensuring the continued growth of proficient programmers.

Bibliography

- [1] “Java,” Internet: https://java.com/en/download/faq/whatis_java.xml, Access July 7, 2020.
- [2] S. I. Ao et al. ed., IAENG Transactions on Engineering Sciences - Special Issue for the International Association of Engineers Conferences 2016 (Volume II), World Sci. Pub., pp. 517-530, 2018. <http://www.worldscientific.com/worldscibooks/10.1142/10727>
- [3] S. S. Wint, N. Funabiki, and M. Kuribayashi, “Design and implementation of desktop-version Java programming learning assistant system,” Proc. HISS, pp. 254-257, Nov. 2018.
- [4] “JUnit,” Internet: <http://www.junit.org/>, Access July 7, 2020.
- [5] K. Beck, *Test-driven development: by example*, Addison-Wesley, 2002.
- [6] “Diary of kencoba,” Internet: <http://d.hatena.ne.jp/kencoba/20120831/1346398388>, Access July 7, 2020.
- [7] N. Ishihara, N. Funabiki, M. Kuribayashi, and W.-C. Kao, “A software architecture for Java programming learning assistant system,” J. Comp. Soft. Eng., vol. 2, no. 1, Sept. 2017.
- [8] N. Funabiki, Tana, K. K. Zaw, N. Ishihara, and W.-C. Kao, “A graph-based blank element selection algorithm for fill-in-blank problems in Java programming learning assistant system,” IAENG J. CS, vol. 44, no. 2, pp. 247-260, May 2017.
- [9] K. K. Zaw, N. Funabiki, and W.-C. Kao, “A proposal of value trace problem for algorithm code reading in Java programming learning assistant system,” Info. Eng. Exp., vol. 1, no. 3, pp. 9-18, Sept. 2015.
- [10] N. Ishihara, N. Funabiki, and W.-C. Kao, “A proposal of statement fill-in-blank problem using program dependence graph in Java programming learning assistant system,” Info. Engr. Exp., vol. 1, no. 3, pp. 19-28, Sept. 2015.
- [11] N. Funabiki, Y. Matsushima, T. Nakanishi, K. Watanabe, and N. Amano, “A Java programming learning assistant system using test-driven development method,” IAENG J. CS, vol.40, no.1, pp. 38-46, Feb. 2013.
- [12] “BFS,” Internet: <http://www.geeksforgeeks.org/breadth-first-traversal-for-a-graph>, Access July 7, 2020.
- [13] “JUnit-Tools,” Internet: <http://www.junit-tools.org/index.php/getting-started>, Access July 7, 2020.

- [14] Y. Higo, A. Saitoh, G. Yamada, T. Miyake, S. Kusumoto, and K. Inoue, "A pluggable tool for measuring software metrics from source code," Proc. IWSM-MENSURA, pp. 2-12, 2011.
- [15] T. G. S. Filó and M. A. S. Bigonha, "A catalogue of thresholds for object-oriented software metrics," Proc. SOFTENG, pp. 48-55, 2015.
- [16] "Metric Plugin," Internet: <http://metrics.sourceforge.net>, Access July 7, 2020.
- [17] "Standard Streams," Internet: https://en.wikipedia.org/wiki/Standard_streams, Access July 7, 2020.
- [18] D. M. Ritchie, "A Stream Input-Output System", AT&T Bell Lab. Tech. J., 68(8), Oct. 1984.
- [19] "Exception Handling in Java," Internet: <https://www.javatpoint.com/exception-handling-in-java>, Access July 7, 2020.
- [20] M. Takahashi, Easy Java, 5th Ed., Soft Bank Creative, 2013.
- [21] H. Yuuki, Java programming lessen, 3rd Ed., Soft Bank Creative, 2012.
- [22] Y. D. Liang, Introduction to Java programming, 9th Ed., Pearson Education, 2014.
- [23] "Java programming seminar," Internet: <http://java.it-manual.com/start/about.html>, Access July 7, 2020.
- [24] "Kita Soft Koubo," Internet: <http://kitako.tokyo/lib/JavaExercise.aspx>, Access July 7, 2020. .
- [25] M. J. Garbade, "Top 3 most popular programming languages in 2018," <https://hackernoon.com/top-3-most-popular-programming-languages-in-2018-and-their-annual-salaries-51b4a7354e06>, Access July 7, 2020.
- [26] S. Cass, "The 2015 Top Ten Programming Languages - IEEE Spectrum," Internet: <https://www.linkedin.com/pulse/2015-top-ten-programming-languages-ieee-spectrum-farzin-pashae>, Access July 7, 2020.
- [27] S. Cass and P. Bulusu, "Interactive: The Top Programming Languages 2018," Internet: <https://spectrum.ieee.org/static/interactive-the-top-programming-languages-2018>, Access July 7, 2020.
- [28] N. Funabiki, K. K. Zaw, N. Ishihara, and W.-C. Kao, "Java programming learning assistant system: JPLAS," IAENG Trans. Eng. Sci., Spec. Issue. Int. Assoc. Eng. Conf. 2016 vol. II, pp. 517-530, 2016
- [29] N. Funabiki, Y. Matsushim, T. Nakanishi, K. Watanabe, and N. Amano, "A Java programming learning assistant system using test-driven development method," IAENG Int. J. Comput. Sci., vol. 40, no. 1, pp. 38-46, Feb. 2013.
- [30] K. K. Zaw, N. Funabiki, E. E. Mon, and W.-C. Kao, "An informative test code approach for studying three object-oriented programming concepts by code writing problem in Java programming learning assistant system," Proc. Global Conf. Consum. Elect. (GCCE2018), pp. 592-596, Oct. 2018.

- [31] "JCF," Internet: <https://docs.oracle.com/javase/tutorial/collections/>, Access July 7, 2020.
- [32] "Iterators in Java," Internet: <https://www.geeksforgeeks.org/iterators-in-java/>, Access July 7, 2020.
- [33] "Metrics 1.3.6," Internet: <http://metrics.sourceforge.net>, Access July 7, 2020.
- [34] K. K. Zaw and N. Funabiki, "A concept of value trace problem for Java code reading education," Proc. Int. Cong. Adv. Appl. Info., July 2015, pp. 253-258.
- [35] K. K. Zaw, N. Funabiki, and W.-C. Kao, "A proposal of value trace problem for algorithm code reading in Java programming learning assistant system," to appear in Inf. Eng. Exp., 2015.
- [36] J. Brunet, D. Guerrero, J. Figueiredo, "Design Tests: An Approach to Programmatically Check your Code Against Design Rules," Proc. Int. Conf. on SW Eng. Comp. vol. pp. 225-258, January 2009.
- [37] Y. Akahane, H. Kitaya, and U. Inoue, "Design and evaluation of automated Scoring Java programming assignments," Proc. Int. Conf. Soft. Eng., Art. Intel., Net. Para./Dist. Comput., pp.1-6, 2015.
- [38] H. Kitaya and U. Inoue, "An online automated scoring system for Java programming assignments," Int. J. Inform. Edu. Tech., vol. 6, no. 4, pp. 275-279, Apr. 2016.
- [39] C. Fu and B. G. Ryder, "Exception-chain analysis: revealing exception handling architecture in Java server applications," Proc. Int. Conf. Soft. Eng., pp. 230-239, May 2007.
- [40] Z. Zhu, Y. Zou, B. Xie, Y. Jin, Z. Lin, and L. Zhang, "Mining API usage examples from test code," Proc. IEEE Int. Conf. Soft. Mainte. Evo., pp. 301-310, 2014.
- [41] C. Kolassa, M. Look, K. Müller, A. Roth, D. Rei, and B. Rumpe, "TUnit –unit testing for template-based code generators," Proc. Modellierung Conf., pp. 221-236, 2016.
- [42] H. Grönniger, H. Krahn, B. Rumpe, M. Schindler, S. Völkel, "MontiCore: A framework for the development of textual domain specific languages," Proc. Int. Conf. Soft. Eng. (ICSE), 2008.
- [43] H. Krahn, B. Rumpe, S. Völkel, "MontiCore: Modular development of textual domain specific languages," Proc. Int. Conf. Model. Tech. Tool. Comp. Perform. Evaluation, pp. 297-315, 2008.
- [44] H. Krahn, B. Rumpe, S. Völkel, "MontiCore: "A framework for compositional development of domain specific languages," Int. J. SW Tool. Tech. Transfer, vol. 12, no. 5, pp. 353-372, Sept. 2010.
- [45] R. Rashkovits and I. Lavy, "Students' understanding of advanced properties of Java exceptions," J. Info. Tech. Edu., vol. 11, pp. 327-352, 2012.
- [46] S. Nakshatri, M. Hegde, and S. Thandra, "Analysis of exception handling patterns in Java projects: an empirical study," Proc. IEEE/ACM Work. Conf. Mining Soft. Rep., pp. 500-503, May 2016.

- [47] J. Bloch, *Effective Java*, 2nd Ed., Prentice Hall PTR, Upper Saddle River, NJ, USA, 2008.
- [48] T. Xue, S. Ying, Q. Wu, X. Jia, X. Hu, X. Zhai, and T. Zhang, "Verifying integrity of exception handling in service-oriented software," *Int. J. Grid. Utility Comput.*, vol. 8, pp. 17-21, 2017.
- [49] R. J. S. Júnior and R. Coelho, "Preserving the exception handling design rules in software product line context: a practical approach," *Proc. Latin-American Symp. Depend. Comp. Work.*, pp. 9-16, 2011.
- [50] G. Fairbanks, D. Garlan, and W. Scherlis, "Design fragments make using frameworks easier," *Proc. OOPSLA Conf.* vol. 41, pp. 75–88, Oct. 2006.
- [51] K. Koile and D. Singer "Development of a Tablet-PC-based System to Increase Instructor-Student Classroom Interactions and Student Learning," *Proc. WIPTE (Workshop on the Impact of Pen-based Tech. on Edu., Purdue University)* Apr. 2006.
- [52] E. Zhou, Z. Niibori, S. Okamoto, M. Kamada, and T. Yonekura, "IslayTouch: an educational visual programming environment for tablet devices", *J. Space-Based and Situated Comput.*, vol. 6, no. 3, pp. 183-197, 2016.
- [53] N. Yamamoto, "An improved group discussion system for active learning using smart-phone and its experimental evaluation," *J. Space-Base. Situated Comput.*, vol. 6, no. 4, pp. 221-227, 2016.
- [54] B. Robinson, M. D. Ernst, J. H. Perkins, V. Augustine and N. Li, "Scaling up automated test generation: Automatically generating maintainable regression unit tests for programs," *26th IEEE/ACM Int. Conf. on Auto. SW Eng. (ASE 2011)*, pp. 23-32, 2011.
- [55] T. Kakeshita and M. Murata, "Application of Programming Education Support Tool pg-tracer for Homework Assignment," *J. of Learn. Tech. and Learn. Envr*, vol1. no.1, pp. 41-60, Mar.,2018.
- [56] "Standard Streams," Internet: https://en.wikipedia.org/wiki/Standard_streams, Access July 7, 2020.
- [57] S. Heckman, "An Empirical Study of In-Class Laboratories on Student Learning of Linear Data Structures," *Proc. Int. Conf. Int Comp. Edu. Res.* pp. 217-225, Aug. 2015.
- [58] M. G'omez-Albarr'an, "The teaching and learning of programming: a survey of supporting software tools," *Comput. J.*, vol. 48, no. 25, pp. 130-144, 2005.
- [59] P. Brusilovsky and S. Sosnovsky, "Individualized exercises for self-assessment of programming knowledge: An evaluation of QuizPACK," *ACM J. Edu. Res. Comput.*, vol. 5, no. 3, pp. 1-22, Sep. 2005.
- [60] M. Kordaki, "A drawing and multi-representational computer environment for beginners learning of programming using C: Design and pilot formative evaluation," *Comput. Edu.*, vol. 54, pp. 69-87, 2010.
- [61] M. J. Lee and A. J. Ko, "Personifying programming tool feedback improves novice programmers learning," in *Proc. ICER*, pp. 109-116, Aug. 2011.

- [62] F. W. B. Li and C. Watson, "Game-based concept visualization for learning programming," in Proc. MTDL, pp. 37-42, Dec. 2011.
- [63] W.-Y. Hwang, R. Shadiev, C.-Y. Wang, and Z.-H. Huang, "A pilot study of cooperative programming learning behavior and its relationship with students learning performance," Comput. Edu., vol. 58, pp. 1267-1281, 2012.