JPEE 실천공학교육논문지
JOURNAL OF PRACTICAL ENGINEERING EDUCATION

실천공학교육 특별호
J. Pract. Eng. Educ. 16(1), 11-19, 2024

Analysis and Application of Front-End Code Playground Tools for Web Programming Education

Aaron Daniel Snowberger[1], Semin Kim[1], SungHee Woo[2]*

[1]Department of Computer Education, Jeonju National University of Education 55101, Korea
[2]Department of Computer Engineering, Korea National University of Transportation, Chungju 27469, Korea

[Abstract]

Web programming courses are often included in university Computer Science programs as introductory and foundational computer programming courses. However, amateur programmers often have difficulty learning how to integrate HTML, CSS, JavaScript, and various preprocessors or libraries to create websites. Additionally, many web programming mistakes do not produce visible output in the browser. Therefore, in recent years, Front-End Code Playground (FECP) tools that incorporate HTML, CSS, and JavaScript into a single, online web-based application have become popular. These tools allow web coding to happen directly in the browser and provide immediate visual feedback to users. Such immediate visual feedback can be particularly beneficial for amateur coders to learn and practice with. Therefore, this study gathers data on various FECP tools, compares their differences, and provides an analysis of how such tools benefit students. This study concludes with an outline of the application of FECP to web programming courses to enhance the learning experience.

Key Words: Front-end, Web Programming, Programming Education, JavaScript, Code Playgroun

I. Introduction

The Internet has become an integral part of human life, encompassing everything from enhancing social connections and promoting social dialogues, to the growth and proliferation of businesses both online and offline, to widespread online education and the rapid distribution of scientific knowledge. Websites that are designed and developed well, with intuitive interfaces and useful features have been the primary drivers of the growth and innovation of the Internet. Therefore, teaching effective web design and development remains an important necessity in many Computer Science programs.

In recent years, however, the rapid growth in the number of web programming frameworks and libraries has far outpaced the academic world. While HTML, CSS, and JavaScript remain the backbone of web page architecture, integrating numerous tools and libraries into web programming classes remains challenging. Additionally, amateur programmers often find it difficult to learn not only the different syntaxes of HTML, CSS, and JavaScript, but also how they link and work together.

Many mistakes in web programming do not result in any visible output in the browser. This makes learning and debugging more difficult. Therefore, in recent years Front-End Code Playground (FECP) tools that incorporate HTML, CSS, and JavaScript into a single, online web-based applications have become increasingly popular. These tools

http://dx.doi.org/10.14702/JPEE.2024.011

Received 30 November 2023; **Revised** 21 December 2023
Accepted 28 December 2023

***Corresponding Author**
E-mail: shwoo@ut.ac.kr

provide immediate feedback and provide an interactive sandbox environment where web coding languages can be practiced and explored. This can be particularly beneficial for amateur coders to visualize and understand the effect that each keystroke has on the webpage outcome.

II. Related Studies

Research papers on web programming curriculum from the early 2010s detail recommended methods [1] as well as challenges and opportunities [2,3] for teaching web programming. These studies were conducted around when the standard for HTML5.1 was officially drafted (2012), and the standard for CSS 2.1 was officially published (2011). Since then, both HTML5 and CSS3 have become standard fixtures in the web programming world. Although these papers are somewhat dated, they still address many of the problems that students continue to face when first learning web programming. Some of these problems include:

1) needing to learn many programming language-paradigms and technologies in a short time
2) debugging for the web is challenging
3) many common student mistakes produce no output in the browser

To address these problems (and more), this study suggests FECPs as tools to aid student learning and enhance the overall learning experience. Surprisingly, FECPs have not been widely written about in the research literature. A Polish paper from 2018 may be one of the only papers that compared and analyzed numerous FECPs [4]. However, that paper did not analyze how FECPs would be beneficial in a web programming curriculum. Therefore, the remainder of this paper will discuss various FECPs, their differences, and the benefits they provide students (and teachers) in the classroom. The paper will conclude with an overview of the practical application of FECPs to the web programming curriculum.

III. Methods

FECPs are online platforms that not only integrate HTML, CSS, and JavaScript, but often also include several related technologies, such as preprocessors, as well as the ability to import external files or libraries. Behind the scenes, the FECP's JavaScript listens for changes to any of the code windows containing HTML, CSS, or JavaScript. When code in the code window is changed, the FECP's JavaScript analyzes the code and refreshes the output window to make

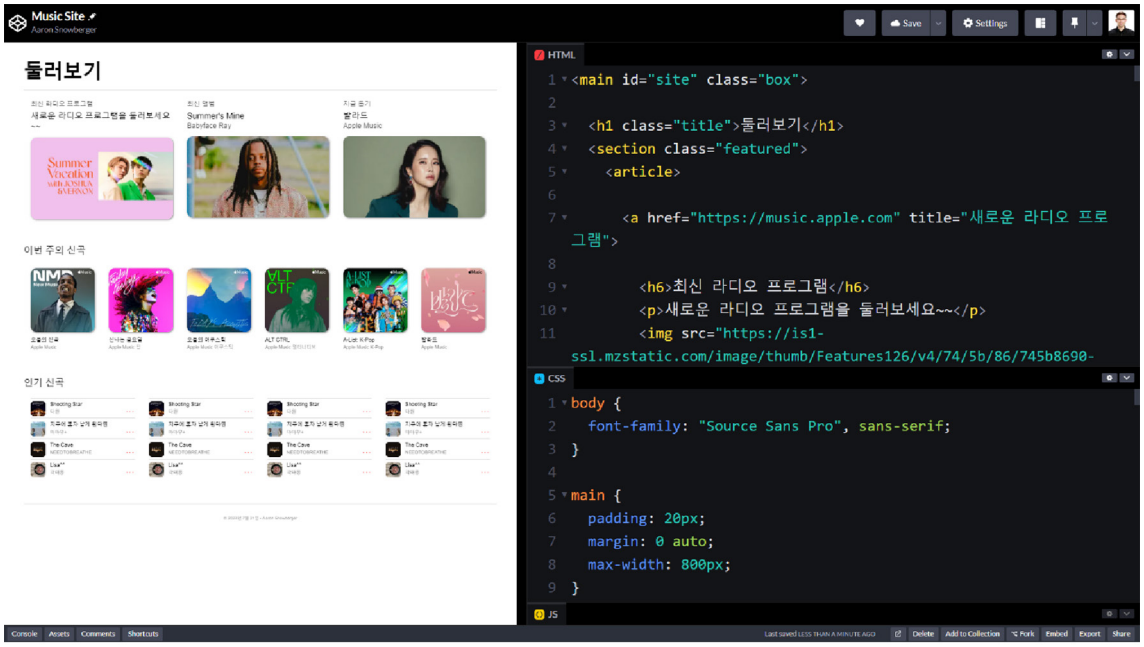

Fig. 1. Example of a webpage built in CodePen.

the results immediately visible. This benefits learners in at least two ways.

First, they can get immediate feedback on the effects of properly written code. Second, because improperly written code produces no visible output, it helps learners to reconsider the code they've just written and how to rewrite it to produce the correct results. Additionally, some of the more advanced FECPs also provide code linting and syntax checking while code is being input. A small 'alert' notification pops up with improperly written code and can also be beneficial for students to immediately double-check what they've just input. Fig. 1 shows an example of a webpage designed in CodePen in a classroom setting.

A. Analysis

This paper reviewed and analyzed eight different FECP platforms and compared their differences. The FECP platforms in this study were chosen based partially on their online popularity, such as web traffic, which was provided by the SimilarWeb traffic analysis Chrome extension [5], as well as online resources that list such tools, such as Wikipedia [6]. These platforms include:

1) CodePen.io [7] (10.2 million monthly visits)
2) JS Fiddle [8] (2 million monthly visits)
3) JS Bin[9] (344 thousand monthly visits)
4) Coder.com[10] (224.2 thousand monthly visits)
5) Plunker[11] (209.1 thousand monthly visits)
6) CSS Deck[12] (121.9 thousand monthly visits)
7) Liveweave[13] (62 thousand monthly visits)
8) JSFeed.io[14] (9.3 thousand monthly visits)

These platforms can be classified into one of three categories based on their level of sophistication and functionality: low-level, mid-level, and high-level functionality. Some common features that each platform includes are code syntax highlighting, multiple coding windows, a Live Preview window, and the ability to import files or libraries from elsewhere on the Internet. The low-level platforms do not include full support for all web languages or have simplified interfaces that lack the code formatting, analysis, and preprocessing features included in mid-level platforms. Mid-level platforms also include more settings, such as font selection, size, and coloring, and window refresh rate, among others. High-level platforms include a 'social coding' aspect with a library of open source, publicly shared code where users can search, share, like, fork, and comment on their favorite examples. All platforms include some method to download or export the code as a ZIP file. Table 1 gives a brief comparison of the platforms analyzed in this study.

Two notes should be made about the FECPs analyzed above. First, Coder.com is a "self-hosted remote development platform" which means that although it can perform the same in-browser coding functionality that the other FECPs can, it first requires installation into a self-hosted environment. This makes Coder.com impractical for classroom use. Second, although CSS Deck has been a very popular code playground in the past, after December 20, 2023, the platform has continually produced 522 server errors.

Table 1. A comparison of FECP platforms

Platform	Monthly Visitors	Distinction	Access	Collaborative	Embeddable	Social (Code Library)	Other
CodePen.io	10.2M	High-level	Free & Paid	Pro Feature	Yes	Yes	HTML, CSS, JS preprocessor support
JS Fiddle	2M	Mid-level	Free	Yes	Yes	No	HAML, CSS, JS preprocessor support
JS Bin	344K	Mid-level (low)	Free & Paid	No	No	No	HTML, CSS, JS preprocessor support
Coder.com*	224.2K	Mid-level	Free & Paid	Yes	Yes	No	Non-web language support included. Requires installation.
Plunker	209.1K	Mid-level	Free	Yes	No	Yes	TypeScript, JS library support
CSS Deck	121.9K	Mid-level	Free	No	No	Yes	Offline late 2023
Liveweave	62K	Low-level	Free	Yes	No	No	No preprocessors
JSFeed.io	9.3K	High-level	Free & Paid	Yes	Yes	Yes	HTML, CSS, JS preprocessor support

From among the FECPs analyzed, both CodePen.io and JSFeed.io were classified as high-level because they include all the collaborative, embeddable, social, and preprocessor support that were determined to be high-level features. CodePen.io, launched in 2012, is one of the oldest FECPs and is currently the most popular. JSFeed.io, launched in 2022, is the most recently created FECP, and although currently the least popular, SimilarWeb's traffic ranking tool indicates that it is growing quite rapidly.

The other FECPs lack one or more of these features and are therefore classified as mid-level or low-level platforms. Liveweave was classified as a low-level platform for its lack of features, particularly the lack of any kind of preprocessor support. Additionally, although JS Bin lacks many of the high-level features that were checked, its support of multiple HTML, CSS, and JavaScript preprocessors bumped its classification to mid-level, albeit on a lower tier than the other mid-level FECPs.

The following three tables, Tables 2, 3, and 4 list the different preprocessors available on each FECP. This type of language preprocessor integration in FECPs, without a complicated build process, helps to enable the inclusion of newer technologies in web programming classes.

Table 2. HTML preprocessors that are available on different FECP platforms

Platform	HTML preprocessors			
	Markdown	Haml	Slim	Pug (Jade)
CodePen	O	O	O	O
JS Fiddle	X	O	X	X
JS Bin	O	X	X	O
Coder.com	Dependent on Self-hosted Environment			
Plunker	X	X	X	X
CSS Deck	X	O	X	O
Liveweave	X	X	X	X
JSFeed.io	O	O	X	O

B. Application in Education

In general, FECPs enable rapid prototyping and experimentation when learning web programming concepts.

Table 3. CSS preprocessors that are available on different FECP platforms

Platform	CSS preprocessors						
	Less	SCSS	Sass	Stylus	Myth	PostCSS	Autoprefixer
CodePen	O	O	O	O	X	O	O
JS Fiddle	X	O	O	X	X	O	X
JS Bin	O	O	O	O	O	X	X
Coder.com	Dependent on Self-hosted Environment						
Plunker	X	X	X	X	X	X	X
CSS Deck	O	O	O	O	X	X	X
Liveweave	X	X	X	X	X	X	X
JSFeed.io	O	O	O	O	X	X	X

Table 4. JavaScript preprocessors that are available on different FECP platforms

Platform	JavaScript preprocesors					
	CoffeeScript	LiveScript	TypeScript	Babel	Traceur	ClojureScript
CodePen	O	O	O	O	X	X
JS Fiddle	O	X	O	O	X	X
JS Bin	O	O	O	O	O	O
Coder.com	Dependent on Self-hosted Environment					
Plunker	X	X	O	X	X	X
CSS Deck	O	X	X	O	X	X
Liveweave	X	X	X	X	X	X
JSFeed.io	O	O	O	O	X	X

This enables learners to immediately begin coding with what they are learning and aids the learning process by effectively removing (or postponing) the more confusing and difficult work of linking files or setting up a build environment.

Additionally, the open-source libraries of many FECPs contain numerous examples of effective and stylish web code which enables students to easily search for a certain component and learn from that code visually. The ability to 'fork,' or copy, a code example also allows students to modify and experiment with the code on their own. Thus, learners can practice both the code examples presented in textbooks, as well as learn from professional code examples. This kind of 'social coding,' or code sharing, along with other collaborative tools available on high-level FECPs also provides an easy way for instructors to gather and grade student assignments. Instructors only need to collect CodePen IDs from students to have access to all their work.

Additional features of many FECPs include 1) the ability to format code, which helps students learn to match opening and closing tags or curly braces; 2) the ability to fold and unfold code, which also helps with tag matching, as well as helping to focus only on a specific portion of code at one; and 3) code analysis tools, that perform linting such as syntax and variable checking, which helps students quickly pinpoint problem areas in their code and debug them. Often learners do not know specifically where to look to identify problems in their code, so these types of code formatting, linting, and analysis tools are highly beneficial. Figures 2, 3, and 4 display an example of the code analysis tools in each code window in CodePen.

This paper was written in conjunction with two Web Programming Fundamentals courses in Fall 2023. Each course consisted of over 30 students and CodePen was selected as the FECP.

The first four weeks of the course focused on learning the basic structuring elements of a webpage using HTML tags. After each lecture in which new HTML tags were introduced, the students practiced live coding with the instructor in CodePen. The primary goal of the HTML section of the course was to familiarize students with matching opening and closing HTML tags and properly indenting code to create an organized HTML document.

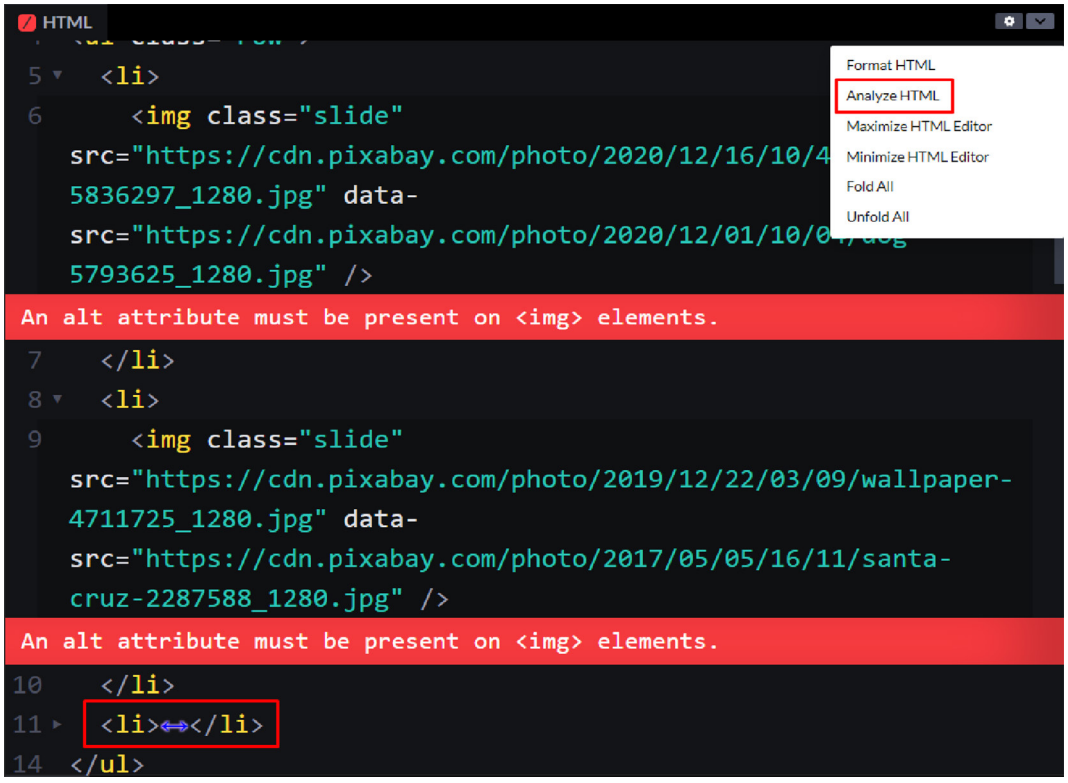

Fig. 2. Example of the HTML analysis tool in CodePen.

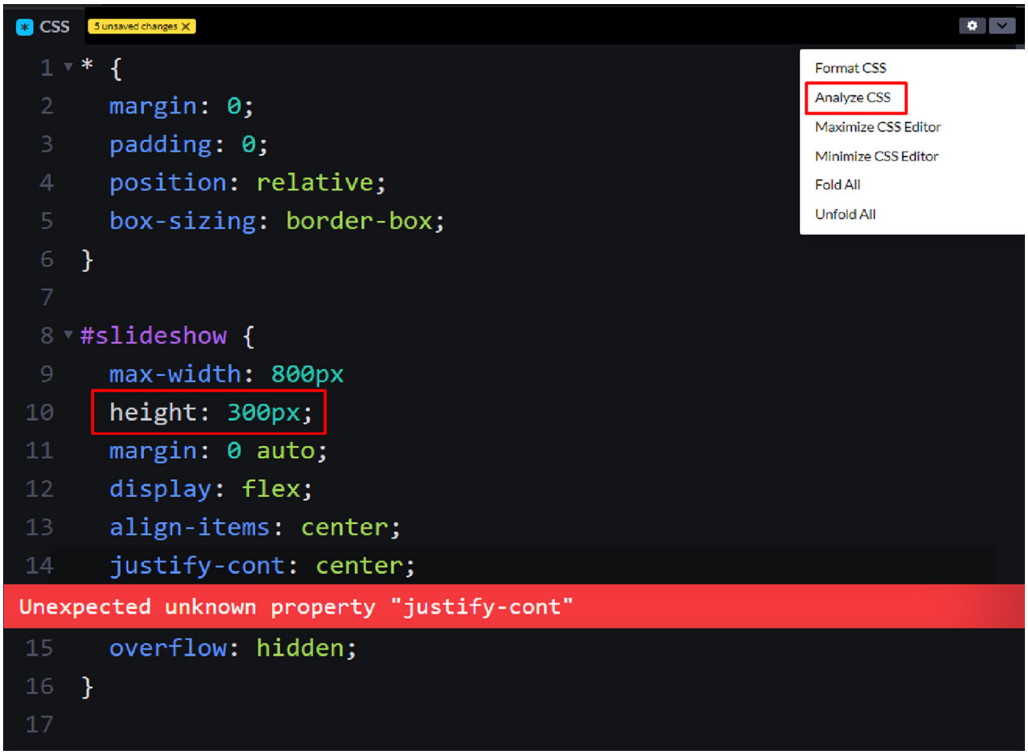

Fig. 3. Example of the CSS analysis tool in CodePen.

Fig. 4. Example of the JavaScript analysis tool in CodePen.

To that end, the Format HTML, Analyze HTML, and code folding tools were introduced and students were presented with various individual coding challenges to ensure the course content was learned effectively.

The following six weeks of the course focused on adding styles to the HTML elements such as colors, layouts, and effects using CSS. The primary goal of the CSS section of the course was to provide students with a variety of options for styling HTML elements. This included the use of inline styles, CodePen's built-in internal stylesheet (in the CSS code window), and importing external CSS files. To that end, CSS reset and normalize were introduced along with various external CSS libraries, such as Bootstrap. Additionally, the concept of web fonts was introduced and both Google Fonts and icon fonts like FontAwesome were incorporated. In CodePen, external CSS files can be directly searched for and imported into a pen's settings. This process helps students understand the concept of importing and managing multiple external files and libraries.

The final five weeks of the course focused on adding interactive functionality to a webpage using JavaScript. The primary goal of the JavaScript section of the course was to introduce students to the concept of the DOM tree and how JavaScript can be used to manipulate it by adding, editing, or deleting DOM nodes. To that end, various simple projects were coded in class that put much of JavaScript's language features to use. These projects included a simple multiplication table using for loops, a slideshow and lightbox using event listeners, and a simple Todo app that created, edited, and deleted DOM nodes entirely in JavaScript.

Due to the limitations of time, only vanilla JavaScript was introduced in these courses. However, CodePen and similar FECPs also offer support for larger JavaScript projects that use front-end libraries such as React, Angular, and Vue. The fact that these libraries can be used without a build process or package manager such as yarn or npm makes FECPs an effective tool to slowly introduce even more advanced front-end coding concepts and paradigms. If time allows in the course, the JavaScript section is also a good time to introduce the various preprocessors available in the FECP as well as variations or replacements for JavaScript, such as syntax for ES5 and beyond, TypeScript, CoffeeScript, and so on. Table 5 gives a brief overview of the key points that were emphasized and related FECP features that were introduced in each section of the course.

IV. Results

The analysis and application of FECPs as described in this paper shows how valuable these tools can be when used in a web programming curriculum. As described above, the primary benefits of using a FECP for teaching web programming fundamentals are as follows.

First, the all-in-one environment allows students to quickly learn and code in three languages at once: HTML, CSS, and JavaScript without the need for managing multiple files. Additionally, as many FECPs include support for preprocessors in each language, instructors can introduce newer coding techniques, paradigms, and languages without needing to rely on maintaining a complicated build process or package manager. Since no setup is required for the coding environment, students can focus their full attention on the coding itself.

Second, FECPs provide immediate feedback on code as it is entered into the coding windows. Additional formatting

Table 5. Overview of a course structure that incorporates FECPs in class

Section	Objectives	Applied knowledge	FECP features introduced
HTML	Matching opening / closing tags Proper indentation of code	Creating web page elements	Format HTML tool Analyze HTML tool Code folding feature
CSS	Various options for styling code Concept of external stylesheets	Modifying colors & images Introduction of web fonts Creating webpage layouts Adding transitions & effects	CSS reset & normalize Importing external stylesheets & libraries
JavaScript	Manipulating DOM elements Events, listeners, & functions Creating & deleting nodes	Functional programming (multiplication tables) Event loop & listeners (slideshow) Single-page Web Apps (Todo)	Importing external script libraries & frameworks Preprocessors (optional)

and analysis tools help students quickly identify problems in their code and debug them. These features are highly beneficial for students who are new to coding because they often do not know where to begin looking for problems in malfunctioning code.

Finally, the collaborative tools and social aspects of many FECPs allow students to learn from other coders, including professional web developers. Students can search a FECP's massive library of functional and stylish web code, 'fork' the code into their own account to edit and learn from it on their own, or share their own user ID link with others. A student's user ID link can also be used by course instructors to have access to all the student's coursework for the class.

V. Conclusion

This paper has focused on a comparison of different FECP platforms and how the use of these platforms can enhance practical training and educational outcomes in web programming courses. The free features and open-source code libraries in the high-level FECPs can aid both instructors and students in achieving learning objectives. Additionally, the progressive application of more complex programming concepts and challenges along with matching FECP tools and features that can be applied to these coding challenges can simplify web programming education as well as motivate students to explore web programming in more depth on their own. Finally, the code highlighting, linting, and analysis features in FECPs directly address two of the major challenges students face when learning programming: 1) that code they write produces no visible outcome, and 2) that they often don't know where to begin looking to debug their code. Therefore, it is recommended that FECPs be incorporated in some aspect into web programming curriculum, or at least be actively recommended to students for learning and experimenting with on their own.

References

[1] S. Xinogalos and T. Kaskalis, "The challenges of teaching web programming: literature review and proposed guidelines," In 8th International Conference on Web Information Systems and Technologies (WEBIST 2012), Porto, Portugal, pp. 207-212. DOI: 10.5220/0003960902070212.

[2] X. Wang and J. C. McKim, "The opportunities and challenges to teach web programming in computer science Curriculum CS2013," *Journal of Computing Sciences in Colleges*, vol. 29, no. 2, pp. 67-78, December 2013. https://dl.acm.org/doi/10.5555/2535418.2535428.

[3] X. Wang, "Design, develop and teach the second web programming course in computer science curriculum," *Journal of Computing Sciences in Colleges,* vol. 29, no. 4, pp. 52-59, April, 2014.

[4] M. Magier and B. Pańczyk, "Comparative analysis of front-end code playground tools," *Journal of Computer Sciences Institute,* vol. 9, pp. 328-333, December 2018. DOI: 10.35784/jcsi.705.

[5] Similarweb – Traffic Rank & Website Analysis, (Google Chrome Extension), https://chromewebstore.google.com/detail/similarweb-traffic-rank-w/hoklmmgfnpapgjgcpechhaamimifchmp.

[6] Wikipedia, "Comparison of online source code playgrounds," [Online]. Available: https://en.wikipedia.org/wiki/Comparison_of_online_source_code_playgrounds#Online_web_client-side_source_code_playgrounds

[7] CodePen – Online Code Editor and Front-End Web Developer, [Online]. Available: https://codepen.io/.

[8] JS Fiddle – Code Playground, [Online]. Available: https://jsfiddle.net/.

[9] JS Bin – Collaborative JavaScript Debugging, [Online]. Available: https://jsbin.com/.

[10] Coder – Your Self-Hosted Remote Development Platform, [Online]. Available: https://coder.com/.

[11] Plunker – Helping you build the web, [Online]. Available: https://plnkr.co/.

[12] CSSDeck – Online HTML, CSS, and JS Code Editor (Sandbox), [Online]. Available: http://cssdeck.com/.

[13] Liveweave – HTML, CSS, and JavaScript demo, [Online]. Available: https://liveweave.com/.

[14] JSFeed – Online Code Editor, [Online]. Available: https://jsfeed.io/.

Aaron Daniel Snowberger_Regular members

2006: Bachelor of Science, Dept. of Computer Science, University of Wyoming
2011: Master of Art, Media Design, Full Sail University
2024: Ph.D., Dept. of Information & Communication Engineering, Hanbat National University
2010 – 2023: Professor, Dept. of Liberal Arts, Jeonju University
2023 – present: Lecturer, Dept. of Computer Education, Jeonju National University of Education
〈Research interests〉 Computer Vision, Deep Learning, Natural Language Processing, Language Education

Semin Kim_Lifetime Membership

2006: Master of Science, Dept. of Computer Education, Woosuk University
2009: Ph.D. Candidate, Dept. of Computer Education, Kongju National University
2018: Ph.D., Dept. of Information & Communication Engineering, Hanbat National University
2020: Master of Science, Graduate School of Sports Science, Hoseo University
2008 – present: Lecturer, Dept. of Computer Education, Jeonju National University of Education
〈Research interests〉 Computer Education, SW/AI Education, Sports Information Systems

SungHee Woo_Lifetime Membership

1993: Master of Science, Dept. of Computer Engineering, Chungbuk National University
1999: Ph.D., Dept. of Computer Engineering, Chungbuk National University
1995 – present: Professor, Dept. of Computer Engineering, Korea National University of Transportation
〈Research interests〉 Information Security, Computer Networks, SW/AI Education