User Experience Evaluation in a Code Playground

Ricardo Queirós
CRACS - INESC-Porto LA, Portugal
uniMAD - ESMAD, Polytechnic of Porto, Portugal

Mário Pinto
uniMAD - ESMAD, Polytechnic of Porto, Portugal

Teresa Terroso
uniMAD - ESMAD, Polytechnic of Porto, Portugal

Abstract

Learning computer programming is a complex activity and requires a lot of practice. The viral pandemic that we are facing has intensified these difficulties. In this context, programming learning platforms play a crucial role. Most of them are characterized by providing a wide range of exercises with progressive complexity, multi-language support, sophisticated interfaces and automatic evaluation and gamification services. Nevertheless, despite the various features provided, others features, which influence user experience, are not emphasized, such as performance and usability. This article presents an user experience evaluation of the LearnJS playground, a JavaScript learning platform which aims to foster the practice of coding. The evaluation highlights two facets of the code playground: performance and a usability. In the former, lab and field data were collected based on Google Lighthouse and PageSpeed Insights reports. In the later, an inquiry was distributed among students from a Web Technologies course with a set of questions based on flexibility, usability and consistency heuristics. Both evaluation studies have a twofold goal: to improve the learning environment in order to be officially used in the next school year and to foster the awareness of user experience in all phases of the software development life-cycle as a key facet in Web applications engagement and loyalty.

2012 ACM Subject Classification Applied computing → Computer-managed instruction; Applied computing → Interactive learning environments; Applied computing → E-learning

Keywords and phrases programming learning, code playground, programming exercises, user experience

Digital Object Identifier 10.4230/OASIcs.ICPEC.2021.17

Category Short Paper

Funding This paper is based on the work done within the Framework for Gamified Programming Education project supported by the European Union's Erasmus Plus programme (agreement no. 2018-1-PL01-KA203-050803).

1 Introduction

The JavaScript language is no more confined to the browser. Nowadays the language gravitates on almost all platforms, assuming a crucial role in distributed and embedded systems. This success came to underpin its use not only by IT companies but also in the first years in computer science related courses.

In this context, LearnJS emerged as a playground for practicing JavaScript coding. There are several playgrounds worldwide. Many of them adopting the paradigm one-size-fits-all paradigm with sophisticated interfaces and offering a large number of exercises with progressive complexity and covering all the typical topics within the curricula of a computer programming course with multi-language support. In order to engage users and accelerate

their extrinsic motivation, some of these platforms include evaluation services with automatic feedback and gamification elements such as leaderboards, achievements, badges and points.

Despite all these features, there are others that aren't enhanced or, sometimes, even neglected such as those that influence more directly the user experience. Therefore, the main requirement in the design of LearnJS was to foster the user experience (UX) in the perspective that having happy users benefit their involvement and, consequently, their loyalty. In order to achieve this requirement, at the platform design phase, two facets of the user experience were addressed: performance and usability. The former was enhanced with the creation of the playground as a Singe Page Application with a small number of round trips to the server. The later, was considered by adopting a UI (User Interface) framework in order to build rich and engaging user experiences composed by crafted components from the Material Design specification.

Despite all these precautions, it is necessary to assess whether they have had a positive effect on the user experience. This article presents a performance and an usability study in the LearnJS playground. The performance study was based on lab and field data obtained from Google Lighthouse and PageSpeed Insights tools. The second is based the perception that users had on the use of the playground. In order to collect these perceptions a usability inquiry was delivered to first- and second-years students of the Degree in Technologies and Information Systems at ESMAD (Superior School of Media Arts and Design) from the Polytechnic Institute of Porto (P.PORTO). The inquiry's questions were based on Nielsen's heuristics such as flexibility, usability and consistency.

These studies aim to achieve two objectives: to improve the user experience on LearnJS, and to raise the awareness of all the workflow agents in the software life-cycle development about the importance of UX in the development of enjoyable virtual learning platforms.

The rest of the article is structured as follows: in section 2 a brief state of the art on code playgrounds and heuristics evaluation models are presented. Section 3 presents LearnJS, more precisely, its architecture, main components, and the playground GUI. In section 4, two types of evaluation in the LearnJS playground were made: a performance evaluation based on Google performance tools reports and an usability evaluation based on the analysis of a survey filled by students of a Web Technology course. Finally, the main contributions of this work are presented and the next steps regarding the evolution of the playground are shared.

2 Code Playgrounds

A variety of code playgrounds have appeared over the years. These type of tools help students to practice a programming language without the need to install any specialized editor or IDE in their computers. The next subsections present a brief survey on code playgrounds and highlight the heuristic evaluation as the main method for evaluating the usability of interactive system such as these types of playgrounds.

2.1 State of the Art

Nowadays, there are several Web applications aiming to foster coding skills. These applications are often coined with several names such as interactive online courses, massive open online course, coding bootcamp tools, code playgrounds, code interview prep platforms, and many others.

Regardless of the names, all of these platforms have something in common – they all encourage the practice of coding. Since solving exercises and receiving feedback on their resolution is considered to be one of the most efficient way to learn programming, these

types of tools are important strategies for students to autonomously consolidate all the programming topics in an informal, enjoyable and interactive way.

Within this scope there are several tools that stand out such as freeCodeCamp, Coursera, codeacademy, edX, Udemy, Udacity, W3schools, SoloLearn, Hackr.io, coderbyte, codewars, hackerrank, codingame, khan academy, qvault, treehouse, lynda, futurelearn, codeasy, zenva academy, and others.

There are several comparative studies [4, 5, 6] analyzing the differences between these tools using several predefined criteria. One of such studies [6], authors compare these code playgrounds based on set of predefined criteria such as edition, evaluation, gamification and interoperability.

In addition to the common objective, many of these platforms share a similar graphic structure that can be grouped into the following components:

- collection of exercises - list of exercises organized by modules and with increasing complexity. In some environments the availability of modules and/or exercises depends on the success in solving the previous exercises;
- statement - description of the challenge in several formats and languages. Beyond the challenge, the text often includes a set of example input/output tests that the student can use for self-evaluation purposes;
- editor - specialize editor embedded in the Web page and properly configured for the supported languages. The main features are syntax highlighting, auto-completion, snippets injection and presentation of syntactical errors;
- console - enumeration of errors, warnings, hints in order to notify students on their actions;
- tests - static and dynamic tests. The former is the base for analyzing the code without running it and detecting many issues from keyword detection to applied code style. The later is based on input/output tests where the student's code is executed with the given input and the output generated upon execution is, then, compared with the given output;
- related resources - dependence resources that are suggested to the students to consolidate the current topic (e.g. similarly exercises, video tutorials);
- social - this component gathers several features that allow the interaction with the community such as forums, code sharing facilities, pair programming, and many others;
- gamification - game elements aiming to motivate students on their progress in the course. They can be materialized as several widgets such as a ranking, a set of badges, or even experience points earned by the students that can unblock levels, permissions or be used later for internal trading.

2.2 Heuristic Evaluation

Heuristic Evaluation (HE) is one of the most widely used methods for evaluating the usability of an interactive system.

Since the time that Schneiderman [8] presented his well-known "Eight Golden Rules of Interface Design", and passing for the popular Nielsen's Ten general principles for interaction design [3] or Tognazzini's First Principles of Interaction Design [9], there were several studies presenting new sets to help UI designers, developers and other experts in their goal of enhancing usability. Despite the number, most of these studies boil down to modifying Nielsen's list and/or adding new principles to evaluate specific aspects not covered. A complete review of several sets of usability heuristics created for specific domains (not only programming) by different authors can be found in [7].

Nowadays, almost everybody refers to evaluate usability or UX based on the Nielsen's heuristics list. Recent studies use Nielsen's or Tognazinni's lists as the most accurate models [1]. Other studies present hybrid solutions joining both models with the premise that Nielsen's list needed something more specific to be useful [1].

In our case, we decided to take Nielsen's list to do this present work. According to Nielsen the practical acceptability of a system includes factors such as usefulness, cost, reliability and interoperability with existing systems. The usefulness factor relates the utility and usability offered by the system. Utility is the capacity of the system to achieve a desired goal. As the system perform more tasks, more utility he has. Usability is defined by Nielsen as a qualitative attribute that estimates how easy is to use an UI. He mentions five characteristics involved in the concept of usability:

- ease of learning - the system should be easy to learn so that the user can start doing some work with the system;
- efficiency - the system should be efficient to use, so after the user learns the system, a high level of productivity is possible;
- memorability - the system should be easy to remember so that the casual user is able to return to the system after a period without using it, without requiring to learn it all over again;
- errors - the system should prevent the user from committing errors as should deal with them gracefully and minimizing the chance of occurring catastrophic errors;
- satisfaction - the system should be pleasant to use so that users become subjectively satisfied when using.

3 LearnJS

LearnJS is a Web playground which enables anyone to practice the JavaScript language [6]. The LearnJS Playground is a Web-based component which will be used by learners to browse learning activities and interact with the compound resources. Here students can see videos or PDF's of specific topics and solve exercises related with those topics with automatic feedback on their resolutions.

Currently, the playground has two main components:

1. Editor: allows students to code their solutions in an interactive environment;
2. Evaluator: assess the student's code based on static and dynamic analyzers.

For the Editor component, the playground uses Ace (maintained as the primary editor for Cloud9 IDE) which can be easily embedded in any web page and JavaScript application. The editor is properly configured for the JavaScript language and supports the Emmet toolkit for the inclusion of dynamic JavaScript snippets. Ace editor can display errors on the editor itself but does not handle language dependencies. A parser needs to be used to detect errors and determine their positions on the source file. There are several tools that can improve code quality. One of such cases is code linters. Linters (e.g JSLint, JSHint) can detect potential bugs, as well as code that is difficult to maintain. These static code analysis tools come into play and help developers spot several issues such as a syntax error, an unused variable, a bug due to an implicit type conversion, or even (if properly configured) coding style issues. LearnJS uses JSHint to accomplish this behavior. While static code analysis tools can spot many different kinds of mistakes, they can not detect if your program is correct, fast or has memory leaks. For that particularly reason, LearnJS combines JSHint with functional tests (based on test cases). For this kind of tests, and since the code is written in JS and the context is the browser, we use a simple approach by iterating all the case tests and

applying the eval function for tests injection. Both analyzers (linter and Test Case runner) are subcomponents of the LearnJS evaluator component that runs exclusively on the client side. This approach avoids successive round-trips to the server which affects negatively the user experience.

At this moment, a simple running prototype (version 0.7.7) is available at `https://rqueiros.github.io/learnjs/`. Figure 1 shows the front-end GUI of the playground.

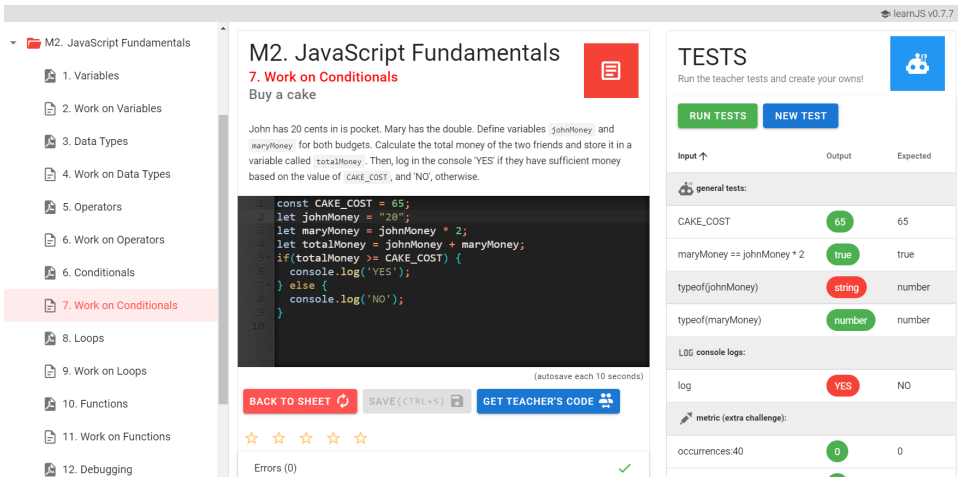

Figure 1 LearnJS playground GUI.

4 User Experience Evaluation

This section presents the two types of evaluation that were performed in the LearnJS playground: performance and usability.

4.1 Performance Evaluation

Nowadays, Web performance is not just a technical concern: it affects everything from accessibility to usability. Even if you look to the software development workflow, design decisions should be aware of their performance implications. There are a lot of examples that a single second spared in the page load of a company site can increase search engine traffic, sign-ups and even the average annual revenue. Thus, it is unanimous that performance is an important factor to bear in mind and it should be measured, monitored and refined continually.

However, the growing complexity of the web poses new challenges that make performance relative. For instance, a site might be fast for one user (on a fast network with a powerful device) but slow for another user (on a slow network with a low-end device) or even a a site might appear to load quickly but respond slowly to user interaction.

So when talking about performance, it's important to be precise and to refer to performance in terms of objective criteria that can be quantitatively measured. These criteria are known as metrics. There are a lot of metrics from the first byte received by the browser to the time when the page is ready to be interactive. Also there are a lot of tools to measure those metrics. In order to overcome this dispersion, Google launched the Web Vitals initiative to simplify the tools and metrics landscape, and help developers focus on the metrics that

matter most. In this realm, tools that assess Web Vitals compliance should consider a page passing if it meets the targets at the 75th percentile for all of the following three metrics:

- Largest Contentful Paint (LCP): measures the time from when the page starts loading to when the largest element is rendered on screen;
- First Input Delay (FID): measures the time from when a user first interacts with your site (e.g. when they click a link, tap a button) to the time when the browser is actually able to respond to that interaction;
- Cumulative Layout Shift (CLS): measures the cumulative score of all unexpected layout shifts that occur between when the page starts loading and when its life-cycle state changes to hidden.

These metrics are generally measured in one of two ways: in the lab using tools to simulate a page load in a controlled environment; in the field on real users actually loading and interacting with the page.

In order to evaluate the LearnJS playground performance, a field analysis was performed using the Google PageSpeed Insights (PSI). Table 1 presents the performance results for six metrics (including the three Web Vitals metrics) organized in the two well known target devices: mobile and desktop. Although the results are presented for both platforms, for the LearnJS scope, desktops are the most important as they will be those used by students in solving programming exercises.

Table 1 LearnJS performance report.

	FCP[1]	SI[2]	LCP	TTI[3]	TBT	CLS
Mobile	7.3s	7.3s	12.3s	9.1s	450ms	0.011
Desktop	1.5s	1.5s	2.1s	1.7s	60ms	0.001

[1] FCP = First Contentful Paint

[2] SI = Speed Index

[3] TTI = Time To Interactive

PSI incorporates data from the Chrome User Experience Report (CrUX) to display data on the actual performance of a page. This data represents an aggregate view of all page loads over the previous 28-day collection period. However, since LearnJS was put online very recently, CrUX doesn't have enough real speed data for this source. Therefore, data comes from the Google LightHouse that gives performance insights from a controlled environment using simulated throttling.

PSI classifies field metric values as good/needs improvement/poor by applying specific thresholds. The overall score for LearnJS was 81 points (0 to 100). It falls in the "needs improvement" category. Looking now at the mobile metric values and stressing the Web Vitals metrics, one can conclude the following:

- LCP (2.1s) - this value was expected since in the SPA paradigm most of the resources are load in the first request. In order to mitigate this score, we will improve it by removing resources that are blocking the main thread of the JS engine in the browser. Also, it is crucial to remove unused JS and CSS files and perform lazy loading in images outside the browser viewport. With these operations we estimate to reduce this value in 1.08s and, thus, it is expected to reach the minimum threshold for the good category (<2000ms);
- TBT (60ms) - it is a replacement for the field-only FID metric. The current value is in the good category range (0 - 100ms). This value is obtained since in the implementation phase of LearnJS we take into account several tasks such as reducing JS execution time

and, consequently, minimizing its main-thread work. Also the reducing of third-party code by replacing large JavaScript libraries with smaller alternatives such as the JSHint and the infinite loop detection have a big impact in the final score;

- CLS (0.001) - the visual stability of LearnJs is very good (between 0 - 0.1) since we adhere to the good practices regarding this metric. One important factor was to flag the image dimensions in HTML in order to the browser reserve space prior to make the respective request. This way, later content is positioned in is final location affecting positively this metric.

4.2 Usability Evaluation

This section evaluates the usability of the graphical user interface of LearnJS based on the Nielsen's model [2], considered the most used usability heuristics for user interface design, analyzing factors such as usefulness and reliability. The usefulness factor relates the utility and usability offered by the system. The reliability is the ability of a system or component to perform its required functions under stated conditions for a specified period. The questionnaire was organized in 13 groups of questions, 12 regarding usefulness (visibility, compatibility, freedom, consistency, error prevention, recognition, flexibility, aesthetic, users help, help and documentation, ease of use and speed) and 1 for the reliability factor, with a total of 51 questions. Each group ended with a request for comments and the inquiry ends by asking to globally classify the LearnJS code playground. The questionnaire was distributed through an online survey, and disseminated to first- and second-year students of a Web Technologies degree at ESMAD, P.PORTO, which represent the target users of LearnJS. The respondents were informed that the questionnaire was anonymous, ensuring confidentiality. A total of 27 responses were gathered.

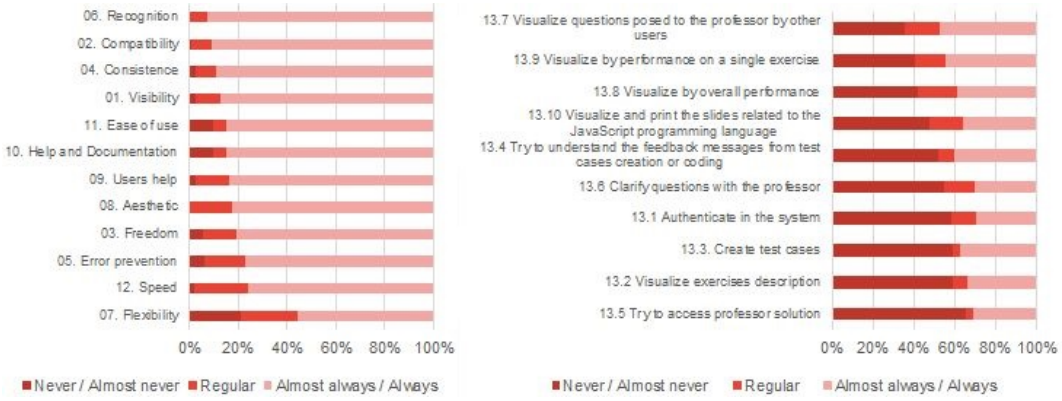

Figure 2 Results of the usefulness (left) and reliability (right) heuristics.

The results showed that recognition, compatibility, and consistence are the aspects most valued by students, as more than 90% of students recognize these characteristics present always or almost always on the platform. Visibility, ease of use and documentation are also aspects highly valued by students. On the other hand, error prevention, speed and flexibility are the aspects, addressed in the questionnaire, less well evaluated by the students. The reliability section intended to check on what tasks students had difficulties, being the most reported ones the visualization of questions posed to the professor by other users, and overall and single exercise performance visualisations. Asked how they would classify the LearnJS playground, given all the parameters previously discussed, 96% of students rated

the platform as good or very good. Globally, some of the respondents' comments indicated the platform was very good, easy to use and learn, well structured and intuitive. On the other hand, some comments suggested the possibility of the platform to provide both basic and advanced solutions for the proposed problems.

5 Conclusion

This paper presents an user experience evaluation performed in the LearnJS code playground. The evaluation was focused on two important facets: performance and usability. The former was based on the lab and field data reported by the Google Lighthouse and PageSpeed Insights tools. The later was based on the user target perception collected in an online survey based on the Nielsen's heuristics.

Since LearnJS was designed keeping in mind these facets, the results obtained ware more or less expected and demonstrates the importance of the awareness of these facets in the design phase of an interactive Web application. Nevertheless, the results pointed some issues that should be solved in the next version of the playground such as flexibility. In this topic students felt no liberty to customize the GUI (e,g, change panels location, alter the background of the editor), to schedule frequent actions (e.g. configure notifications) or the nonexistence of shortcut keys to perform the most used functions (e.g. run tests).

As future work, LearnJS will be capable to be integrated in an e-learning ecosystem based on an Learning Management System (e.g. Moodle, Sakai, BlackBoard). This integration will be based on the Learning Tools Interoperability (LTI) specification. This specification provides several interfaces for the authentication and grading services from/to the LMS (the tool consumer) and the LearnJS (the tool provider). Beyond integration, LearnJS will use in a near future a gamification engine to gamify the learning activities with level unblocking, rankings, experience points, badges and other gamification elements.

References

1 Toni Granollers. Usability evaluation with heuristics, beyond nielsen's list. In *ACHI 2018, The Eleventh International Conference on Advances in Computer-Human Interactions*, pages 60–65, 2018. URL: https://www.thinkmind.org/articles/achi_2018_4_10_20055.pdf.

2 J. Nielsen. *Usability engineering*. Academic Press, 1993. URL: https://books.google.pt/books?id=fnvJ9PnbzJEC.

3 Jacob Nielsen. 10 usability heuristics for user interface design, 1995. URL: https://www.nngroup.com/articles/ten-usability-heuristics/.

4 José Carlos Paiva, José Paulo Leal, and Ricardo Queirós. Game-Based Coding Challenges to Foster Programming Practice. In Ricardo Queirós, Filipe Portela, Mário Pinto, and Alberto Simões, editors, *First International Computer Programming Education Conference (ICPEC 2020)*, volume 81 of *OpenAccess Series in Informatics (OASIcs)*, pages 18:1–18:11, Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. doi:10.4230/OASIcs.ICPEC.2020.18.

5 José Carlos Paiva, José Paulo Leal, and Ricardo Queirós. Fostering programming practice through games. *Information*, 11(11), 2020. doi:10.3390/info11110498.

6 Ricardo Queirós. Learnjs - A javascript learning playground (short paper). In *7th Symposium on Languages, Applications and Technologies, SLATE 2018, June 21-22, 2018, Guimaraes, Portugal*, pages 2:1–2:9, 2018. doi:10.4230/OASIcs.SLATE.2018.2.

7 Daniela Quiñones and Cristian Rusu. How to develop usability heuristics: A systematic literature review. *Computer Standards & Interfaces*, 53:89–122, 2017. doi:10.1016/j.csi.2017.03.009.

8 Ben Shneiderman. *Designing the User Interface: Strategies for Effective Human-Computer Interaction*. Addison-Wesley Longman Publishing Co., Inc., USA, 3rd edition, 1997.

9 B. Tognazzini. First principles, hci design, human computer interaction (hci), principles of hci design, usability testing. `http://www.asktog.com/basics/firstPrinciples.html`. Accessed: 2021-03-28.