LearnJS - A JavaScript Learning Playground

Ricardo Queirós
CRACS & INESC-Porto LA & DI/ESMAD/P.PORTO, Porto, Portugal
ricardoqueiros@esmad.ipp.pt
https://orcid.org/0000-0002-1985-6285

Abstract

The JavaScript ecosystem is evolving dramatically. Nowadays, the language is no longer confined to the boundaries of the browser and is now running in both sides of the Web stack. At the same time, JavaScript it's starting to play also an important role in desktop and mobile applications development. These facts are leading companies to massively adopt JavaScript in their Web/mobile projects and schools to augment the language spectrum among their courses curricula.

Several platforms appeared in recent years aiming to foster the learning of the JavaScript language. Those platforms are mainly characterized with sophisticated UI which allow users to learn JavaScript in a playful and interactive way. Despite its apparent success, these environments are not suitable to be integrated in existent educational platforms. Beyond these interoperability issues, most of these platforms are rigid not allowing teachers to contribute with new exercises, organize the existent exercises in more suitable and modular activities to be deployed in their courses, neither keep track of student's progress.

This paper presents LearnJS as a simple and flexible platform to teach and learn JavaScript. In this platform, instructors can contribute with new exercises and combine them with expositive resources (e.g videos) to define specific course activities. These activities can be gamified with the injection of dynamic attributes to reward the most successful attempts. Finally, instructors can deploy activities in their educational platforms. On the other hand, learners can solve exercises and receive immediate feedback on their solutions through static and dynamic analyzers. Since we are in the early stages of implementation, the paper focus on the presentation of the LearnJS architecture, their main components and their data and integration models. Nevertheless, a prototype of the platform is available in a GitHub repository.

2012 ACM Subject Classification Software and its engineering → General programming languages, Applied computing → Interactive learning environments

Keywords and phrases Web development, programming, e-learning, automatic evaluation

Digital Object Identifier 10.4230/OASIcs.SLATE.2018.2

Category Short Paper

Funding FourEyes is a Research Line within project "TEC4Growth – Pervasive Intelligence, Enhancers and Proofs of Concept with Industrial Impact/NORTE-01-0145-FEDER-000020" financed by the North Portugal Regional Operational Programme (NORTE 2020), under the PORTUGAL 2020 Partnership Agreement, and through the European Regional Development Fund (ERDF).

1 Introduction

Nowadays, the JavaScript (JS) language is no longer seen as a browser scripting language to validate forms and make AJAX calls to Web servers. In fact, the language has evolved in a consistent way and can already be used to create applications on the most popular

7th Symposium on Languages, Applications and Technologies (SLATE 2018).
Editors: Pedro Rangel Henriques, José Paulo Leal, António Leitão, and Xavier Gómez Guinovart
Article No. 2; pp. 2:1–2:9
OpenAccess Series in Informatics
OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

platforms. One of the great impulses for this growth was the appearance of Node.js, which allows developers to use JS throughout the stack (front and backend) of a Web application. But it's not just on the Web domain that JS is having a huge success. In fact, JS can already be used to create native/hybrid mobile (e.g. React Native, Ionic) and desktop applications (e.g. Electron, WinJS, NW.js). Last but not least, several game engines based on JavaScript can be used for making HTML5 games for desktop and mobile web browsers, supporting Canvas and WebGL rendering (e.g Phaser, Cocos2d).

Obviously, the rise of JS and its corresponding omnipresence led companies to start adopting the language since it allows their development teams the need to master a single language for the cross-platform development of their products. This growth has also re-activated the JavaScript community, being nowadays considered one of the most popular languages according to several studies[1]. At the same time, there is a concern from Schools to adjust their courses curricula to teach these skills not only at the language level, but also to adopt the most popular frameworks and tools that are now gravitating on the Web.

In this context, several online platforms have appeared in recent years aiming to foster the learning of JavaScript. These platforms, typically coupled in online learning platforms (e.g Udemy, Udacity), provide sophisticated UI and a very strong level of interaction, facilitating the progress of students through creative examples. Regardless of their popularity, these platforms have issues regarding interoperability with educational systems and flexibility in content management. For instance, teachers can only advise the use of such tools for training purposes and cannot use them to define specific learning activities and keep track on the evolution of students.

This paper presents LearnJS as a learning environment for the teaching-learning process of the JavaScript programming language. The platform allows two main use cases: teachers can contribute with new resources, combine existing resources into activities and distribute activities in learning management systems; students can access activities, solve exercises and receive automatic feedback. Both use cases have important points that should be emphasized. In the case of teachers, the activities created can include expository resources (e.g. PDF, videos) and evaluative resources (e.g. exercises). Also, gamification attributes (e.g. levels, hints, achievements, leaderboards, unlock levels and code skeletons) can be assigned to provide playful and engaged activities to students. In the case of the students the feedback returned by the platform is not only produced by dynamic evaluation (tests cases), but also by static code analysis through the use of linters which are responsible for the inspection of potential buggy code.

The remainder of this paper is organized as follows: Section 2 reviews the existing environments to learn JavaScript and focuses its attention on Web platforms. In this context several platforms are compared according to several criteria: interoperability, flexibility. In Section 3, the LearnJS architecture and its main components are presented. In this context, we expose the data and interoperability models. In Finally, we conclude with a summary of the main contributions and perspectives of future work.

2 Related Work

Learning computer programming can be a lonely, complex, and demotivating process [1, 3, 5]. These issues have been addressed in the last years, with the appearance of several on-line learning environments trying to leverage coding education and make it accessible to everyone,

[1] `https://www.tiobe.com/tiobe-index/`

Table 1 MOOCs features comparison.

	Path	Resources (expositives)	Resources (evaluation)	Social
EdX	Yes	Videos	Quiz	F/D
Coursera	Yes	Videos	Quiz/Puz	UP/F/D
Udacity	Yes	Videos	Quiz	Forum
CodePlayer	No	Videos	No	Com/Rec
CodeAcademy	Yes	ICE	ICE	Ach/Badges
Code.org	Yes	Videos	ICE/Puz	Levels
TreeHouse	Yes	Videos	ICE/Puz	Badges
CodeSchool	Yes	Videos	ICE	F/Badges

even those with absolutely no coding experience or knowledge [7, 4]. These environments come in various formats ranging from non-interactive approaches (e.g. YouTube channels, blogs, books) to integrated and interactive solutions (e.g. intelligent tutors, online coding environments).

Nowadays, there is an enormous demand on the technology sector to be up to date with the latest frameworks and languages. Regardless whether you are a coding newbie or a mature developer, you have several options, besides a computer science degree, to improve your programming skills. In this realm, MOOCs (Massive Open Online Courses) and Online Coding Bootcamps are two increasingly popular options for learners to improve their development skills and find work within a relatively short amount of time. While these two are excellent alternative learning contexts, the two options still have very distinct differences [2].

A MOOC is an online course, usually available without charge, where learners can choose their own learning pace and direction. MOOCs are free educational courses often delivered by renowned university professors that typically feature a mix of downloadable readings, quizzes, discussion boards, video content and peer-to-peer assessment. The goal of MOOCs is to reach a much larger audience than traditional courses can accommodate. Often, MOOCs offer certificates for a fee which are awarded on successful completion of a course, and transferable college credit.

An Online Coding Bootcamp, on the other hand, is an intensive and paid course, usually eight to twelve weeks in duration, which offers hands-on training, career guidance and job assistance. These types of platforms involve a greater time commitment for the learner and are more suitable for those who wants to quickly master a specific language (or stack) and get a technical job.

Table 1 compares a few of the most popular online learning platforms/MOOCs on a set of features and tools.

Most of the MOOCs offer learning paths with a list of courses to work through. This feature is very important, especially for those new to programming. Despite the existence of paths, the studied MOOCs don't offer a very rigid structure allowing learners to choose several learning paths during the course. The materials of the courses come in several flavors and are organized in two types: 1) expository resources, such as videos (the most popular format) and HTML/PDF tutorials and 2) evaluation resources, such as interactive code exercises, quizzes and puzzles. Almost all platforms offer videos as a way to disseminate knowledge. These videos include the resolution step-by-step of exercises. This way, learners gain some theoretical/practical skills that can be later consolidated and applied in coding challenges.

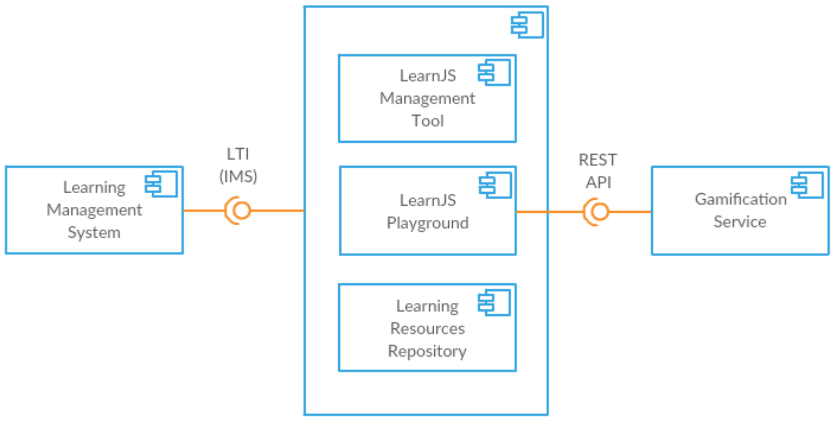


Figure 1 LearnJS component diagram.

These challenges can be run inside of interactive coding exercises components (ICE) giving feedback and support to the learner during the resolution of the exercise (e.g. CodeAcademy, Treehouse). Most of these components are based on cloud IDEs (e.g. Cloud9, Codeanywhere) and integrate some tools like resources sequence, chat and video visualization. Regarding gamification and social features, most platforms adhere to the same components, such as forums (F), learning dashboards (D), user profiles (UP), comments (COM), recommendation (REC), levels and badges. For instance, CodePlayer offers a different approach to learning code by playing code like a video, helping people to learn front-end technologies quickly and interactively. The platform also includes a commenting tool and links to related walkthroughs. CodeAcademy includes a user progress dashboard informing of the current state of the learner regarding its progress in the courses. This platform enhances the participation in the courses by also including achievements (ACH) that are rewarded with badges and users are also able to share completed projects with the rest of the site community and potentially showcase their skills to employers. Except for Code.org, all the platforms have a strong presence in the mobile world, with app versions for Android and iOS.

3 LearnJS

In this section we present LearnJS, a simple and flexible online playground for the teaching and learning of the JavaScript language. The architecture of learnJS is depicted in Figure 1.

At its core, LearnJS is composed by two components used by the two system user profiles:

- **Teachers**: use the **LearnJS Management Tool** to create/select resources to/from the Learning Resources Repository in order to compose a learning activity. Next, they deploy the activity in a Learning Management System.
- **Students**: launch the activity in the LMS and solve it using the **LearnJS Playground**. Beyond the internal gamification features, the playground can benefit from other Gamification Services to foster student's competitiveness and engagement.

The purpose of LearnJS is also to integrate an e-learning ecosystem based on an LMS (e.g. Moodle, Sakai, BlackBoard). For this, it benefits from the interoperability mechanisms to provide authentication directly from the LMS and to submit exercises grades back to the LMS, using the Learning Tools Interoperability (LTI) specification.

In the following sections we detail these two main components in the LearnJS ecosystem: the management tool and the playground.

3.1 LearnJS Management Tool

The LearnJS Management Tool is a Web-based component which will be used by teachers/instructors to submit resources and aggregate them to obtain a composite learning activity. The next section will detail the main aspects of this management tool, more precisely, the GUI component and the resource and activity schemata.

3.1.1 GUI component

The LearnJS Management Tool is a Web-based component based on HTML5 Canvas. Its main purpose is to provide a flexible way for teachers to contribute with new learning resources and allow their aggregation and gamification to define playful learning activities. The final result of this aggregation is a LearnJS manifest with all the necessary information for the correct functioning of the activity in the student's playground.

Another feature of this tool is the capacity for sharing and grading activities. This feature will allow a teacher to share a previously created learning activity in the public space of the LearnJS community. With the grade feature, instructors could score a given activity taken into account the experience that they have with it. This grading will influence the results list after searching.

3.1.2 The resources schema

Teachers can use the LearnJS Management Tool to contribute with new resources. The supported resources in LearnJS follow Sweller and Cooper [6] paradigm based on a learner-centered approach to define a constructivist learning model. This model foster the learning by viewing and learning by doing approaches where educational resources, either expository or evaluative, play a pivotal role. Thus, in LearnJS, resources have two flavours:

- Evaluative: JavaScript challenges to be solved by coding;
- Expositive: Videos or PDF files showing how to master a specific topic.

In this moment, we do not have yet the GUI component finalized. Thus, the submission of a new learning resource should be made through the upload of a JSON file which should comply with the LearnJS official resource schema formalized by a public JSON Schema[2]. The example on Listing 1 shows an evaluative resource for the calculation of a number's factorial.

The JSON document has a simple structure. It contains basic properties for identification and metadata purposes. One of the most important properties is the **type** property. It can assume one of two values: document or exercise. The former requires the **url** property to be set. In this case, the system will load the resource located in that URL. The later requires filling the **exercise** property. This property is composes by the following sub-properties:

- **statement**: the exercise statement formatted in plain text or HTML;
- **hint**: a set of hints to help students to overcome the challenge. By default, they are blocked;
- **code/skeleton**: code skeleton defined by the teacher. Only available by gamification;
- **code/solution**: solution of the challenge submitted by the teacher. Used for input tests injection. Only available after success completion of the exercise by the student;
- **code/tests**: test cases. The input tests are inject in the student's solution and the outputs compared with the provided output tests or with the output generated by the teacher solution.

[2] `https://github.com/rqueiros/learnJS`

Listing 1 Resource JSON instance template file.

```
{
 "id":"http://learnJS/resources/125412",
 "title": "Calculate factorial of a number",
 "url": "",
 "type": "exercise",
 "metadata": {
   "author": "Ricardo Queiros",
   "date": "19-04-2018",
   "level": "intermediate",
   "tags": ["recursivity","math"]
 },
 "exercise": {
   "statement": "Create a function that receives one number and
        returns its factorial",
   "hint": ["Verify special cases like 0 that should return 1"],
   "code": {
     "skeleton": "function factorial(x){ return;}",
       "solution": "function factorial(x){if(x===0){return 1;}
         return x*factorial(x-1);}",
       "tests": [{"in": "4","out": "24"},{"in": "0","out": "1"}]
    } } }
```

.

3.1.3 The activity schema

Teachers can also perform other operations in the management tool, such as the creation of activities.

An activity combines a set of resources of several types (evaluative, expositive) with gamification attributes. Listings 2 shows an activity JSON instance for learning JavaScript arrays.

An activity JSON file is composed by several properties. We highlight two:

- **levels**: can be considered as sub-activities composed by a set of resources identified in the **resources** sub-property. Students should see and solved the respective resources of the level. The completion of the level and the respective unlock of the next level is granted after the student solved a specific percentage of evaluative resources defined in the **perc** property of the level.
- **gamify**: a set of attributes that can be assigned to resources. After a success completion of an evaluative resource, students can be awarded in multiple forms. Hence, the **award** property can have one of the following values:
  - **HintExtra**: gives an extra point to the learner. The learner can spend the hint points on any exercise by unhiding the hint associated;
  - **ShowNextSkeleton|ShowAllSkeleton**: gives the learner the ability to unhide the code skeleton associated to the next (or all) gamified resources;
  - **UnlockLevel|UnlockAllLevels**: gives the learner the ability to unlock the next (or all) level.

Listing 2 Learning activity JSON instance.

```
{
 "id": "http://learnJS/activities/129387",
 "title": "Learn the basics of Arrays",
 "metadata": {
   "author":"Ricardo Queiros",
   "date":"19-04-1975",
   "level":"basic",
   "tags":["arrays"]
 },
 "levels": [
  {"id":"1", "name":"Basic operations", "perc":"75",
          "resources": ["...resources/125412", "..."]},
  {"id":"2", "name":"Sort", "perc":"50", "resources":["..."]}
 ],
 "gamify": [
   {"resource": ".../resources/125412", "award":"HintExtra"},
   {"resource": ".../resources/225232", "award":"ShowNextSkeleton"}
 ]
}
```

3.2 LearnJS Playground

The LearnJS Playground is a Web-based component which will be used by students/learners to browse learning activities and interact with the compound resources. Here students can see videos of specific topics and solve exercises related with those topics with automatic feedback on their resolutions. The architecture of the playground is shown in Figure 2.

The playground is composed by three main components:

1. **Editor**: allows students to code their solutions in a interactive environment;
2. **Evaluator**: assess the student's solution based on static and dynamic analyzers;
3. **Gamification Engine**: gamifies the learning activity with the management of levels and several awards.

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.

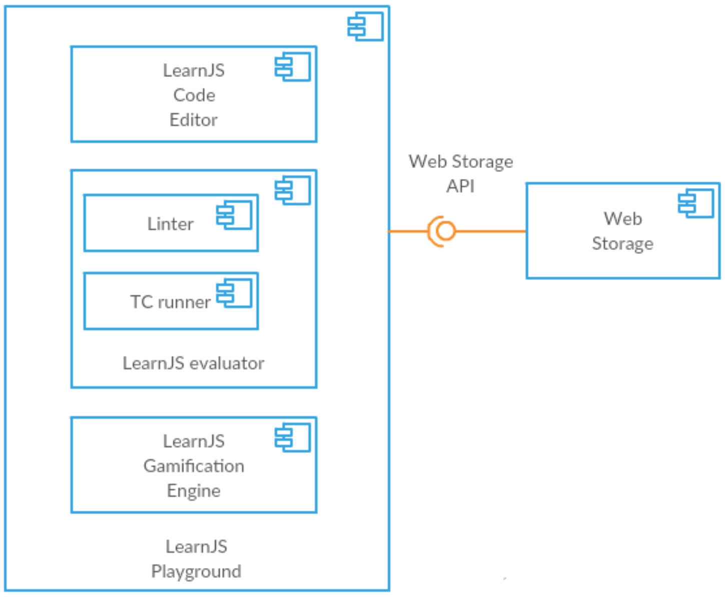


Figure 2 LearnJS Playground component diagram.

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.

Lastly, the **Gamification Engine component** is responsible for loading/parsing the LearnJS manifest and fetching resources from the learning resources store. If levels are defined, the engine sequences and organizes the resources properly. Upon completion of evaluative resources from students, the engine deals with all the logic associated with the respective awards by unhiding/unlocking features of next challenges. Finally, the component send the results back to the server.

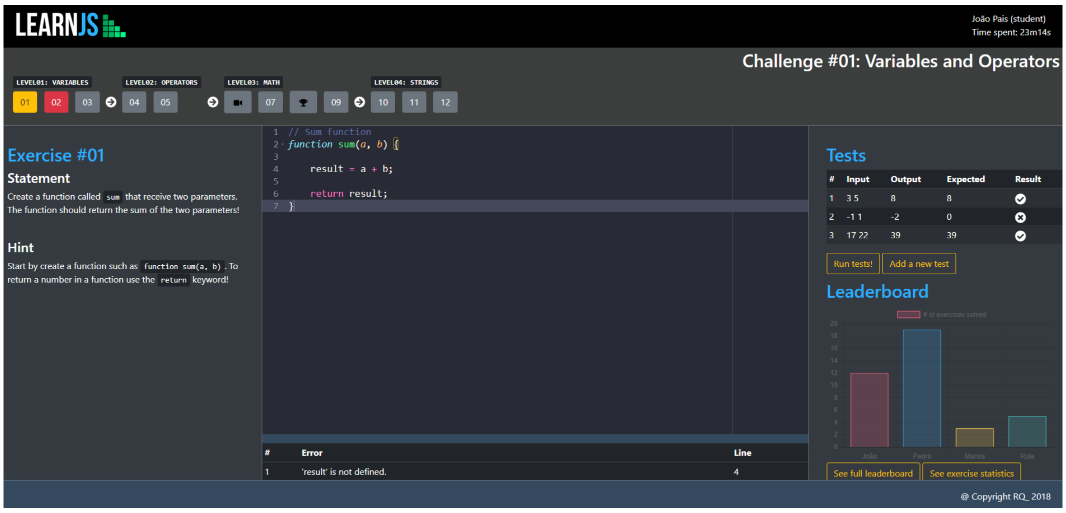
At this moment, we have a simple running prototype. The source code is available at a GitHub repository. Figure 3 shows the frontend GUI of the playground.

4 Conclusions

In this paper we present LearnJS as a flexible playground for JavaScript learning. Since we are in the beginning of implementation, the paper stresses the design of the platform divided in two main components: the management tool and the playground. In the former, teachers can contribute with new exercises and bundle related exercises in learning activities. All these entities were formalized using JSON schemata. The later, allows students through a sophisticated and interactive UI, to see and solve educational resources (mostly, videos and exercises). In order to engage students, the platform can be configured to gamify resources through the subgrouping of activities in levels, the assignment of awards and the exhibition of a global leaderboard.

The main contributions of this work is the design of a platform with interoperability concerns in mind and the respective schemata for the simple concepts of educational resources and activities.

As future work we intend to create a more mature prototype by creating a introductory course for novice students to learn JavaScript. Then, for validation purposes, we intend to



Exercise #01

Statement

Create a function called `sum` that receive two parameters. The function should return the sum of the two parameters!

Hint

Start by create a function such as `function sum(a, b)`. To return a number in a function use the `return` keyword!

```
// Sum function
function sum(a, b) {

    result = a + b;

    return result;
}
```

#	Error	Line
1	'result' is not defined.	4

Tests

#	Input	Output	Expected	Result
1	3 5	8	8	✓
2	-1 1	-2	0	✗
3	17 22	39	39	✓

Leaderboard

Figure 3 LearnJS Playground UI.

use the platform in real classes and receive student's feedback. After this process, our idea is to work on the management tool. Regarding the playground, our intentions is to maintain it very simple, avoid at maximum the communication with the server and improve the game mechanics of the engine.

References

1 Kirsti M. Ala-Mutka. A survey of automated assessment approaches for programming assignments. *Computer Science Education*, 15(2):83–102, 2005. doi:10.1080/0899340050015074 7.

2 Gemma Church. MOOCs versus coding bootcamps. https://www.class-central.com/report/moocs-versus-coding-bootcamps/, 2016. [Online; accessed april 19th, 2018].

3 Jackie O'Kelly and J. Paul Gibson. Robocode & problem-based learning: A non-prescriptive approach to teaching programming. *SIGCSE Bulletin*, 38(3):217–221, 2006. doi:10.1145/1140123.1140182.

4 Pedro Xavier Pacheco and António Coelho. Computer-based assessment system for e-learning applied to programming education. In *4th International Conference of Education, Research and Innovation*, pages 3738–3747, 2011.

5 Anthony Robins, Janet Rountree, and Nathan Rountree. Learning and teaching programming: A review and discussion. *Computer Science Education*, 13(2):137–172, 2003. doi:10.1076/csed.13.2.137.14200.

6 John Sweller and Graham Cooper. The use of worked examples as a substitute for problem solving in learning algebra. *Cognition and Instruction*, 2(1):59–89, 1985. doi:10.1207/s1532690xci0201_3.

7 Elena Verdú, Luisa M. Regueras, María J. Verdú, José P. Leal, Juan P. de Castro, and Ricardo Queirós. A distributed system for learning programming on-line. *Computers and Education*, 58(1):1–10, 2012. doi:10.1016/j.compedu.2011.08.015.