Improving the Authoring of Web-based Interactive E-books with FableJS

Alfredo Tito Silva[1], Welton Marinho de Souza[1], Daniel de Sousa Moraes[1], Roberto Gerson Azevedo[2], Carlos de Salles Soares Neto[1]

[1]Telemídia - Universidade Federal do Maranhão (UFMA)
Av. dos Portugueses s/n - Campus do Bacanga
São Luís/MA - CEP: 65085-580 - Brasil

[2]Ecole Polytechnique Fédérale de Lausanne
Route Cantonale, 1015 Lausanne, Suíça

{alfredtito97, welton457, csallesneto, dacavieira, hedvanfp}@gmail.com

roberto.azevedo@epfl.ch

Abstract. *Electronic books (e-books) first appeared aiming to convert physical books to the digital format. Nevertheless, as technologies advanced, new features were being added to them, creating then, the interactive multimedia e-books. Developing these types of interactive content often requires technical knowledge. However, for authors without this experience, there are few solutions with a high level abstraction paradigm. This article describes FableJS, a library for authoring web applications with nonlinear interactive e-books. Also, a controlled experiment with volunteers using the proposed tool is shown. The study points out that 76.92% of the participants evaluated the library's expressiveness as satisfactory.*

1. Introduction

Electronic books (e-books) were, initially, created with the goal of convert large literary masterpieces into digital format (HART, 1971). With the advancements of the web, however, these books have not only been converted to the digital form, but also gained new features, such as multimedia content and interactivity, leading to the urge of interactive e-books.

By providing enhanced features such as animation, touch, and sound, interactive e-books allow the development of non-linear interactive stories with enhanced and more immersive experience to readers. Developing that sort of story usually requires specific programming knowledge, and due to this, most of the created stories end up not exploring all the available resources.

Domain specific languages are an interesting alternative to general purpose programming languages to the development of interactive e-books. [removed for double blind review 2017], for instance, gathered a set of requirements for the development of interactive e-books and then proposed a conceptual model, named *Interactive E-books Model*[1], supporting these requirements. Despite being a model

[1]*Model's name changed for double blind review

to develop new web applications, a domain-specific tool is required in order to help instantiate this model.

This article presents FableJS, a library designed for the development of non-linear interactive e-books instantiating and improving the *Interactive E-books Model*. The library incorporates components to extend the current functionalities of the model in order to facilitate the customization of the story and reuse of components. In addition, a qualitative experiment used to gather feedback about the usability and expressiveness of the library is also presented.

The rest of the paper is structured as follows. Section 2 discusses related works. Section 3 discusses the proposed extensions to the *Interactive E-books Model*implemented into the FableJS library. Section 4 presents a experiment with users and its results. Finally, Section 5 brings the conclusions and recommendations for future works.

2. Related Work

Regarding e-books, there are two areas in which research is being developed which is new technologies for e-book systems and psychological or educational e-book evaluations. The first focuses on techniques for the implementation of new e-book readers, respectively, the other area aims at the study of book replacement through e-book, e-book usability, and effectiveness of e-books in education.

This work was developed focusing on the first line of research, in addition, the presented related works carried out in the same branch. In [Kazi et al. 2014] presents the Kitty tool that is based on sketches to create dynamic and interactive illustrations, it has a generalized structure, so it's possible to create children's e-books, cartoon strips and more. In its structure it uses a relational graph for communication and effects between objects.

The work in [Choi et al. 2014] demonstrates the developed electronic reader that supports hyperlink, multimedia, user interaction, animation and 3D, as well as portability for the Android mobile system. The system was developed using HTML5 and CSS3 for layout and structuring, while JavaScript was responsible for behavior control. The authors present a tool that allows a rich interaction and visualization using technologies that facilitate the development of e-books.

These presented works contain similarities with the FableJS regarding the use of multimedia, interaction, and technologies used. The main difference is that, this proposed work does not provide a graphic interface. However, FableJS introduces the mechanics of the agents as well as covering a range of multimedia and interactive features not supported by the aforementioned works.

3. FableJS extensions to the *Interactive E-books Model*

This work proposes an improved and extended instance of the *Interactive E-books Model*, with additional elements to enhance its use in the authoring of narratives in electronic books. The proposed library inherits all the elements defined by the *Interactive E-books Model*. Thus, it proposes new elements and attributes to redefine the way these features can be used in the document. The new elements are described in the following subsections.

3.1. Board

The *Interactive E-books Model*allows the use of CSS3 for improvements in the style of the presentation of its elements, however, for users with small knowledge of style rules, this can make the creation process more difficult. In addition to that, the verbosity of the document increases by blending HTML with CSS. The elements **<board>** and **<alert>** were created to abstract the need of using CSS approach directly. Instead, the user only need to declare the element using its attributes.

The **<board>** element works as a board or speech bubble. In addition to eliminating the use of CSS directly in the document, it improves the visibility of overlapping text to a background image not conducive to its reading. It has attributes like set-class, font-size, and color, in order to encapsulate the entire CSS part thus decreasing verbosity. The Listing 1 shows how the element **<board>** is declared using FableJS.

Listing 1. Board using the FableJS element

```
<board id="board1" set-class="classic-board" font-size="20" width="300" color="white">
    Text comes here.
</board>
```

3.2. Alert

The **<alert>** component works as a warning popup. It shows a warning box when its parent component receives an event. It differs from the **<board>** because it is shown only when an event occurs, showing, then, the desired message. Previously, to simulate the same effect using an instance of *Interactive E-books Model*, it was necessary to create an agent just to send the message. The Listing 2 demonstrates the new **<alert>** element offered by FableJS.

Figure 1 demonstrates the use of **<board>** and **<alert>** elements. The board here is the white balloon displaying some text of the story. The alert appears as a grayish frame displayed when the reader clicks on the mushroom.

Listing 2. Using the element alert

```
    <agent id="door">
        <state id="locked">
            <on-touch>
                <img src="assets/door-locked.png">
                <alert>Door is open!</alert>
                <change-state target="unlocked">
                </change-state>
            </on-touch>
        </state>
    </agent>

```

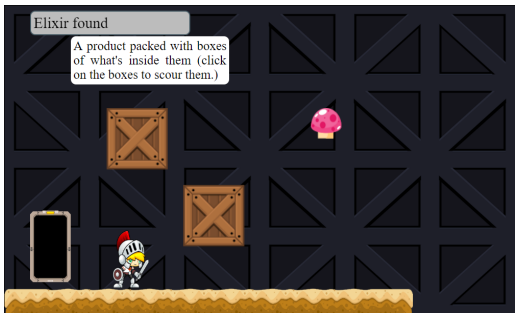

Figura 1. Board and alert example.

3.3. Draggable and Detect

The *Interactive E-books Model*defines the event **drag**, as one of its events. However, in previous instances, there was no implementation of this type of interaction. Thus, FableJs introduces this new feature through the elements **<draggable>** and **<detect>**.

The **<draggable>** attribute has the function of making a media draggable on the screen, whilst the **<detect>** element causes other media to detect the **drag** event of the media with the **<draggable>** attribute. Realizing the ocurrence of the event, the second

media, with the **<detect>** element, generates a state transition in an agent. An example of how this elements work is shown in Listing 3.

Listing 3. Daggrable and Detect example

```
<agent id="key">
    <state id="main">
        <img id="keyTreasure" draggable src="assets/key.png">
    </state>
</agent>
<agent id="treasure-box">
    <state id="locked">
        <img id="assets/box-locked.png" left="400" top="400"/>
        <detect target="keyTreasure" agent="door"
         change-state="unlocked"></detect>
    </state>
    <state id="unlocked">
        <img src="graphics/box-unlocked.png"/>
    </state>
</agent>
```

3.4. Trigger

The **<trigger>** element was created to simplify the **<on-capture>** element that works associated with **<emit>**. In the *Interactive E-books Model*, the **<on-capture>** has the function of detecting events emitted in a story through the element **<emit>** and thus modifying the state of the desired agent.

The Listing 4 shows how to use these elements. When the **<agent>** with the attribute **id** "button-door"changes its state to "lock", it emits the "openBox"event for the page scope. Then, all agents with the **<on-capture>** element, capture the event and verify if it is the same one that they are waiting for. Thus, the **<agent>** "door"that is waiting through **<on-capture>**, receives the event and makes the change of state.

Listing 4. Using the <trigger> element

```
<agent id="door">
  <state id="locked">
    <img src="assets/DoorLocked.png"/>
  </state>
  <state id="open">
    <img src="assets/DoorOpen.png"/>
  </state>
</agent>
<agent id="button-door">
  <state id="unlock">
    <img src="graphics/Switch_unlock.png"/>
    <trigger agent="door" change-state="unlocked">
    </trigger>
  </state>
</agent>
```

3.5. Transitions

The *Interactive E-books Model*also defines transitions effects for media and pages, which also were not implemented in previous instances. Adding this resource, FableJS intends to make the contents of the stories more fluid.

The effects are provided by the Animate.css library, which offers 77 already implemented presentation effects that can be used by web applications. In order to use them in FableJS, it is necessary to add the attribute "transition ="in the tag of the element desired to use the effect, and then set the parameters "animated + effect"as shown in the Listing 5.

Listing 5. Transitions examples

```
<img transition="animated fadeOut" id="box" src="graphics/box.png" />
<img transition="animated infinite tada" src="graphics/rock.png">
```

4. Experiment

This section describes an experiment to investigate how the proposed library behaved in the hands of users without prior knowledge and experience in constructing interactive narratives. The experiment aims to evaluate if the model and the library can express in a simple and intuitive way the elements commonly present in possible e-books.

4.1. Setup

The experiment was carried out with the help of 13 volunteers from an introductory course in Computer Science, and was conducted in a computer lab with a physical structure suitable to accommodate all students of the subject.

4.2. Methodology

The first contact with the participants consisted of a presentation of the *Interactive E-books Model*and the library FableJS. Followed by the distribution of the supporting material, that consisted of examples and an explanatory site with the steps to initiate the development.

After the presentation, the next step was the activity of developing an interactive narrative using the proposed library and the help of the supporting material. From this moment the students were followed for two weeks via web platform (Slack)[2].

4.3. Participants

Data were collected from the participants' profiles, and it was observed that the class consists of 73.3% men and 26.7% women and half of them come from public schools and the other half from private schools. Also, 23.07% are less than 18 years of age and the others (76.92%) are older. The majority of the class (93.1%) had computer science as their first choice in the entrance exam.

Among the participants, 10 of them (76.92%) indicated that they did not have declarative language knowledge, while the other 3 (23.07%) reported knowing HTML. Besides, 11 students reported having little or no ability with web programming and 2 students reported already possessing a certain skill.

About having knowledge of interactive media, 7 students reported that they did not have any knowledge, 5 reported having some knowledge and 1 person reported having a certain domain. This indicates that a considerable portion of the class does not have experience with interactive media or does not master this subject.

In general, it is possible to verify that when starting the Computer science course, the majority of the students still do not have a clear notion on programming and interactive media and do not dominate the technical skill in these areas.

[2]Team collaboration software toolkit and cloud-based online services. https://slack.com/

4.4. Questionnaire for User Satisfaction (QUIS)

The QUIS was based on a previously existing questionnaire to measure user satisfaction with the [Chin et al. 1988] interface. During the development of the tool, one of the concerns was to offer a more appropriate interaction and interface to the users.

To verify this, the following quality criteria were used in the questionnaire: **usability**, **user experience** and **communicability**. It is reported that the questions verified in these aspects were also adapted to the scope of this work and included both the *FablesJS* tool and the *web* site where the information about its use was hosted.

Usability considers the ease of learning of a system and the use of its interface [Barbosa and Silva 2010], whereas user experience considers the user's sense of This communication assists in the satisfaction indication of this user. Communicability is related to interface communication intent and system activity logic, as designed by the developer team and designer.

4.5. Results and discussions

After the two weeks of follow-up of the students in the experiment of the tool we had the answer of 13 participants, being that 7 of them managed to deliver a complete narrative.

4.5.1. Users Impressions

The graph in Figure 2, which represents the user's general impressions section, shows the positivity of volunteer feedbacks, questions 1 and 2 (Qt1 and Qt2), ask if the library presented was considered in Overall good and if it met the needs of the narratives thought out. Where 84.61% of the students found the tool to be excellent and 76.92% found it to be satisfactory to their needs, the students' feedback showed us that the model had a certain quality as well as the library.

On the resources and flexibility of the library, questions 5 and 6 (Qt5 and Qt6) show a 61.53 % satisfaction in the amount of resources offered and 84.61 % versatility of the resources offered.

4.5.2. System Terminology and Information

On the terms and information given to the user during the production of the interactive e-books, in Figure 3, it is observed that we had a lot of neutrality in the answers, mainly regarding the relation of the terms with the task that is being executed and about the instructions for the commands, both with 53.84% neutral responses, tending more towards a positive response, which may be indicative of lack of experience from the public in the experiment.

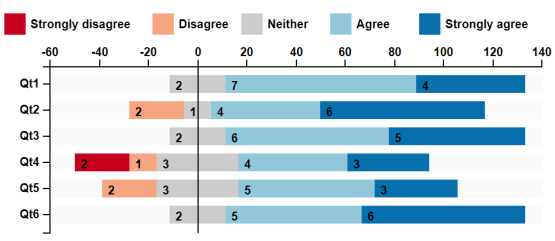

Figura 2. Impressions as user

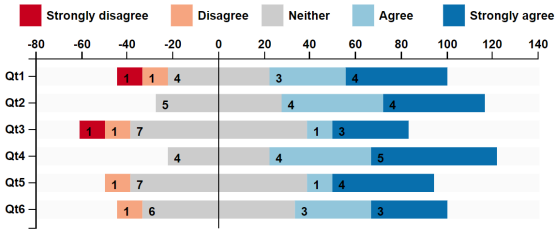

Figura 3. Terminology and System Information

4.5.3. System Learning

Discussing how was the system learning by the students, it is observed in Figure 5, question 2 (Qt2) showing that 61.53% did not have an easy time in starting to build e-books with FableJS. And question 3 (Qt3) shows the difficulty in learning advanced functionalities. However, in questions 1 and 4 (Qt1 and Qt4), 76.92% of the students reported that they had no difficulty in learning to operate the library and that the learning time was not long.

It is also observed in questions 5,6 and 7, a facility to explore functions both by trial and error, as well as, by browsing the documentation provided. This shows that despite the difficulties in the beginning of learning the tool becomes very understandable.

4.5.4. FablesJS Capacity

When asked about the capacity of the FabelJS library, we see in Figure 4 a positive feedback, as very effective (61.53%), in questions 1 and 2 (Qt1 and Qt2), which evaluate the response time to create some element and the system response time as a whole. We also obtained a low error rate (30.76 %) addressed in question 3 (Qt3).

Questions 4, 5 and 6 (Qt4, Qt5 and Qt6) show that (61.53%) said that to operate the library requires some experience. It was also observed that for the majority (61.53%), completing the tasks requires learning a few commands. E (53.84%) said they have a facility in using the available shortcuts and functions. Which leads us to conclude that the functions are in the right quantity and working as expected in several different platforms, but that the experience in the use makes the tasks even simpler.

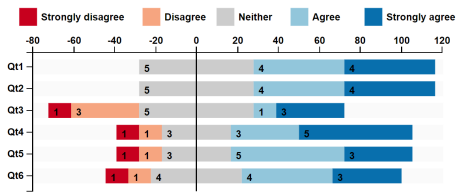

Figura 4. Fables Capacity

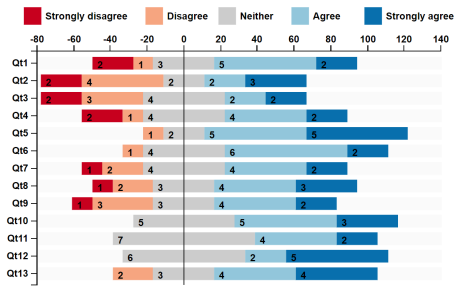

Figura 5. System Learning

4.6. Online tutorial

A set of tutorials and a website with the necessary steps and documentation for the initiation of library use were made available to students to serve as a basis for the production of their work.

The resources available for consultation were evaluated. Figure 6, highlights the result of question 2 (Qt2), which questions whether the language used in the resources was clear and direct. And question 7 (Qt7), that evaluates the usefulness of the available website, with a positive valuation of 76.92%.

The Figure 8 demonstrates how easy students have considered importing the library and utilizing it for playback of interactive e-books, with 61.53% satisfaction in story reproduction and error feedback during development and 84.61% satisfaction in the speed required to install the library.

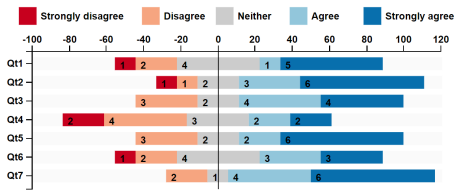

Figura 6. Site evaluation

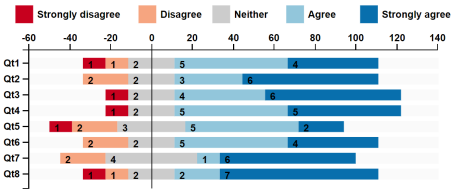

Figura 7. Tutorial and examples evaluation

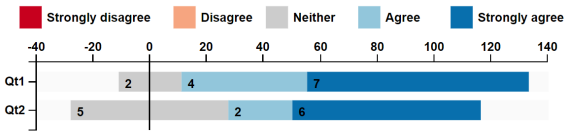

Figura 8. Installation and use evaluation

5. Conclusion

Improvements developed before and after the experiment show that the document instantiating the *Interactive E-books Model*is undergoing improvements, confirmed by users of the tool as they find it easy to use. The developed extensions show that the document instantiating the Fables model is undergoing improvements, confirmed by the users of the tool considering it easy to use. In addition, decreasing verbosity by encapsulating some elements shows that the document has been refined.

As future works, we propose the development of a graphical interface for manipulating the FableJS library while maintaining the visualization of the generated code. Additionally, it is intended to develop the other environments defined by the *Interactive E-books Model*, such as interactive e-books sharing and export of the created applications.

Referências

Barbosa, S. and Silva, B. (2010). *Interação humano-computador*. Elsevier Brasil.

Chin, J. P., Diehl, V. A., and Norman, K. L. (1988). Development of an instrument measuring user satisfaction of the human-computer interface. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 213–218. ACM.

Choi, J., Lee, Y., and Kim, K. (2014). Html5 based interactive e-book reader. *International Journal of Software Engineering and Its Applications*, 8(2):67–74.

Kazi, R. H., Chevalier, F., Grossman, T., and Fitzmaurice, G. (2014). Kitty: sketching dynamic and interactive illustrations. In *Proceedings of the 27th annual ACM symposium on User interface software and technology*, pages 395–405. ACM.

removed for double blind review (2017). Removed for double blind review. In *Proceedings of the 2017 ACM Symposium on Document Engineering*, pages 19–28. ACM.