Enabling the creation of web-first scholarly articles: the RAJE editor

Gianmarco Spinaci
Digital Humanities and Digital Knowledge,
University of Bologna, Bologna, Italy
gianmarco.spinaci
@studio.unibo.it

Silvio Peroni, Angelo Di Iorio, Fabio Vitali
Digital and Semantic Publishing Laboratory,
Department of Computer Science and
Engineering, University of Bologna, Bologna,
Italy
{silvio.peroni, angelo.diiorio,
fabio.vitali}@unibo.it

ABSTRACT

In this demo paper, we introduce the new version of the *RASH Javascript Editor* (RAJE), an open source standalone wordprocessor for writing HTML scholarly articles compliant with the *Research Articles in Simplified HTML* (RASH) format.

Categories and Subject Descriptors

Applied computing [**Document management and text processing**]: Document management—*Text editing*; Applied computing [**Computers in other domains**]: Personal computers and PC applications—*Word processors*

Keywords

RAJE, RASH, Web-first publishing formats, Word processors

1. INTRODUCTION

In the last years, the use of PDF as submission/publication format for scientific articles has been criticised by several people, and several discussions have been raised about using HTML instead - in fact several conferences (e.g. the International Semantic Web Conference) and journals (e.g. Data Science published by IOS Press) are now adopting HTML-based formats for preparing and sharing scholarly works. One of the most used formats is *RASH - Research Articles in Simplified HTML* [2]. RASH restricts the use of HTML elements to only 32 elements, so as to guarantee a better long-term preservability of these documents and to keep the cognitive load of its users affordable. Minimality and a rigorous pattern-based approach are the key features of RASH, compared to other extensions or subsets of HTML proposed by the community towards the same goal (e.g. ScholarlyHTML[1]).

Even if converters from ODT/DOCX documents into RASH are available, one of the main obstacles to its broad adoption was the absence of a well-developed and a multi-platform *what you see is what you get* (WYSIWYG) word processor compliant with RASH natively.

A few months ago we have made available a preliminary release of the *RASH Javascript Editor* (RAJE) [4] to solve this issue. RAJE is an open source standalone wordprocessor designed for being totally compatible with RASH and the other existing tools included in the RASH Framework. We run some test sessions on RAJE that allowed us to collect several comments that showed some crucial flaws in the functionalities and interface. This paper presents the new version of RAJE built upon this feedback.

This new version simplifies the interaction, the interface, and the installation of RAJE. During the demo presentation, we will show all its main features and we will directly involve people in using it. We are also interested in fostering the discussion of the community on HTML-based formats for scholarly articles and to collect more feedback.

The rest of the paper is organised as follows. In Section 2, we introduce some of the most relevant HTML-based word processors. In Section 3, we introduce the new version of RAJE, and we discuss the main differences and added features with respect to the old one. Finally, in Section 4, we conclude the paper sketching out some future works.

2. RELATED WORKS

In the past years, several editors have been proposed for the creation of native HTML documents. In this section, we briefly list some of the most representatives.

Fidus Writer[2] is an open source WYSIWYG collaborative HTML-based word processor made for academics who need to use citations and formulas within papers. All the articles created with it can be exported in many ways: as a website, a paper, or an ebook. The main focus of the editor is on the content of the paper, and the final layout of the paper can be chosen once the content is finalised.

Authorea[3] is an online platform which allows one to write and edit scientific papers by means of a particular Web-based word processor. In fact, each article is an HTML document stored in a particular Git repository. In particular, Git is used to keeping track of the different versions of a document. It uses LaTeX so as to produce the article layout ready for being submitted to academic events. In addition, Authorea allows authors to write mathematical notations, tables, plots and figures either in LaTeX or MathML, and a DOI is automatically assigned to it once it is publicly shared through the platform.

Dokieli[4] is a client-side editor for decentralised article publishing, annotations, and social interactions [1]. Since

[1]`https://github.com/w3c/scholarly-html`

[2]`https://www.fiduswriter.org/`
[3]`https://www.authorea.com`
[4]`https://dokie.li/`

Dokieli follows a pure decentralisation principle, authors can publish the paper where they prefer. In particular, Dokieli is based on Solid[5] so as to allow one to edit the article in the browser and to save it directly to server storage.

3. THE RE-ENGINEERED VERSION OF RAJE

Last year we have released a pre-alpha version of the *RASH Javascript Editor* (*RAJE*) [4]. We tested it involving 6 people in a user testing session, where we asked them to reproduce a particular document by using the editor. While the evaluation of the perceived usability of RAJE was acceptable, the test also highlighted some critical issues of the system. In particular, the users stressed on three weaknesses: (i) lack of some shortcuts and keys to make the navigation smooth, (ii) difficulties in inserting and modifying some content elements and (iii) issues on portability and installation. Thus, we have entirely re-engineered RAJE so as to address these problems.

The new architecture of RAJE is shown in Fig. 1. In particular, it is now defined by means of two specific modules. The first module, i.e. *RAJE-core*[6], implements the main editing functionalities, and it is based on TinyMCE[7] – contrarily, the previous version of RAJE was totally developed from scratch. The other module, i.e. *RAJE-app*[8], is a wrapper that uses RAJE core to implement the standalone wordprocessor. RAJE-app allows one to store RASH documents either in the file system or in online storages such as Github (as of 7 January 2018, still an experimental feature) and Dropbox (not implemented yet). The main idea of this new infrastructure is to keep the core separated from the wrapper separated so as to enable the reuse of the editing capabilities of RAJE in different environments (within a Web application, imported in existing frameworks, etc.) – while, in the previous version, RAJE was a monolithic artefact.

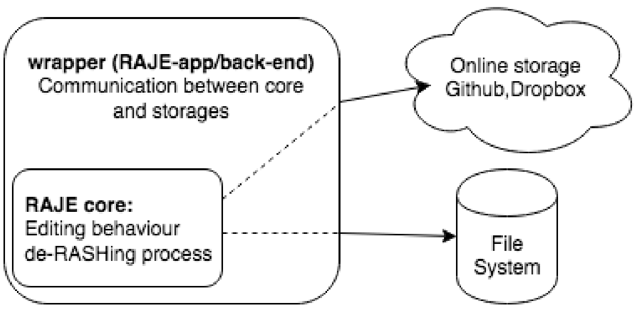


Figure 1: The re-engineered version of the RAJE structure. Core and Wrapper are two different projects. The wrapper integrates the core inside its body and it takes care of the communication between core and the various storages both local and online.

3.1 RAJE core

[5]https://solid.mit.edu/
[6]https://github.com/rash-framework/raje-core
[7]https://www.tinymce.com/
[8]https://github.com/rash-framework/raje-app/

RAJE-core manages everything that concerns the editing of the RASH documents. As introduced in [2], RASH is a rather simple HTML-based markup language, which focuses on representing the content of an article avoiding to repeat the same information and without caring of the specification of presentational objects, such as the label to associate to in-text reference pointers or to any other complex objects (e.g. sections, figures, tables). The rendering of these labels – "Figure 2", "[3]", the number associated to sections, the metadata associated to the paper, etc. – is done automatically by the view package available in RASH and implemented in CSS3+Javascript, which enables to show the whole content of the paper.

This distinction between the original RASH file (*RASH source* from now on) and the way it is visualised in the browser after the application of CSS3 stylesheets and Javascript scripts (*Rash view* from now on) is a peculiar choice of the design of RASH. It was made to improve modularity and flexibility, and to speed up the editing process.

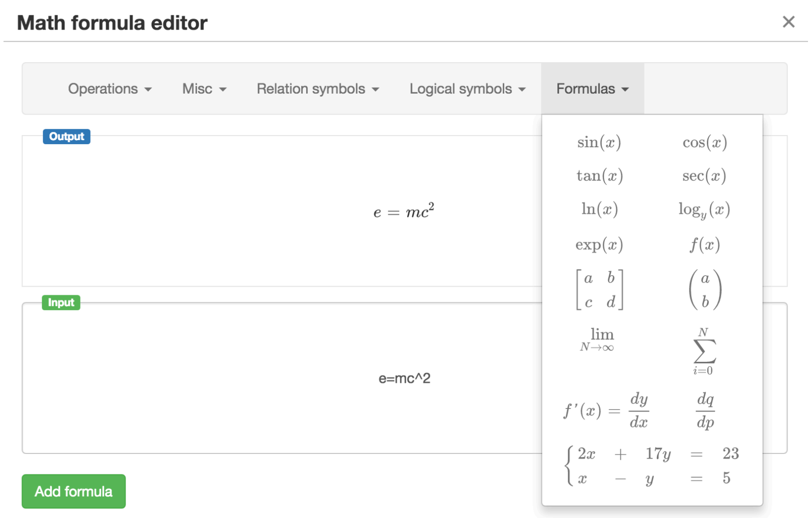
It is also fundamental to understand how RAJE-core works. In fact, RAJE-core addresses two main tasks: the **editing the RASH view** (through its WYSIWYG interface) and the **conversion of the RASH view into the RASH source** (called *de-RASHing process*). These two operations are necessary so as to enable the modification of the document and to prepare the current visualised content to be properly stored according to RASH (source). It is worth mentioning that the storing operation is not implemented by the core since this is designed to only deal with the editing process itself.

The new version of the RAJE core was entirely rewritten and extended on top of TinyMCE. The adoption of TinyMCE has simplified several typical operations such as the undo/redo. This has been implemented by using the TinyMCE undo manager as the basic layer: the RAJE core handles complex undo-redo operation by combining data from the lower layer. There have been several other pros in the adoption of TinyMCE, besides the aforementioned undo/redo handling, among which a better stability in the creation and deletion of inline elements (`em`, `strong`, etc.).

On the other hand, the use of TinyMCE has required us to solve several other issues for handling formulas and footnotes. In RASH, the best and most-Webby choice for defining mathematical formulas is to use MathML, that is rendered by means of MathJax[9]. However, the problem with this approach is that TinyMCE blocks any injected script, including those that are needed by Mathjax for rendering such formulas. Thus, in order to allow RAJE-core to edit formulas, we have developed a modal dialog that has been defined outside TinyMCE. This modal dialog, shown in Fig. 2, allows one to write the formulas in AsciiMath[10] (input panel) and to visualise them in real-time as SVG content (output panel) by means of MathJax.

When a formula is added (by pressing the button *add formula*), the related SVG content is added in the TinyMCE content with an additional attribute, i.e. `data-mathml`, which includes the MathML representation of the formula so as to be used when storing the document. This allows us to easily transform the formula from the SVG format into MathML during the de-RASHing process, obtaining a RASH source

[9]http://www.mathjax.org/
[10]http://asciimath.org/



$e = mc^2$

e=mc^2

$\sin(x)$ $\cos(x)$ $\tan(x)$ $\sec(x)$ $\ln(x)$ $\log_y(x)$ $\exp(x)$ $f(x)$ $\begin{bmatrix} a & b \\ c & d \end{bmatrix}$ $\begin{pmatrix} a \\ b \end{pmatrix}$ $\lim_{N\to\infty}$ $\sum_{i=0}^{N}$ $f'(x) = \frac{dy}{dx}$ $\frac{dq}{dp}$ $\begin{cases} 2x + 17y = 23 \\ x - y = 5 \end{cases}$

Figure 2: The modal dialog implementing the RAJE-core editor for mathematical formulas. This editor also provides a toolbar with the AsciiMath templates for several operations, symbols and formulas.

with only MathML formulas. Similarly, during the start-up of RAJE-core, all the formulas (which are stored in MathML in the RASH source) are temporarily moved in a wrapper element outside TinyMCE, processed in SVG, and then put again in the editor body as SVG content.

Another important difference with the previous version of RAJE is the way the article metadata (title, authors, keywords, etc.) are handled in the new version. In particular, we have developed a modal dialog which is opened once one click on any metadata. In addition, the order of the authors can be changed by dragging and dropping them in the favourite position. All these choices made the interaction with RAJE much easier and comfortable for the final users.

3.2 RAJE-app

The RAJE-app[11] is the wordprocessor that enables one to load, save and modify HTML documents compliant with RASH. Technically speaking, it is a wrapper (as defined in Fig. 1) that reuses RAJE-core and adoptsElectron[12] for developing a standalone application – available for Apple OS X, Microsoft Windows, and Linux. The main responsibility of the RAJE-app is to wrap the core and manage communications between it and other external actors. For instance, it implements the storing of RASH documents by following the process introduced in Fig. 3.

The system uses a similar approach to implement other functionalities as well. For instance, RAJE-app integrates NodeGit[13], i.e. a Node.js module that imports a light git standalone, so as to implement the storing of an article within a GitHub repository. While it would be possible to use directly the GitHub API for addressing this operation (which was the approach used in the old version of RAJE), having a real local git repository works better since a single commit can wrap more files – something that cannot be done by using the GitHub API – and it can also be performed offline. Thus, the use of NodeGit enables RAJE to have an

[11]`https://rash-framework.github.io/raje-app/`
[12]`https://electronjs.org/`
[13]`http://nodegit.org/`

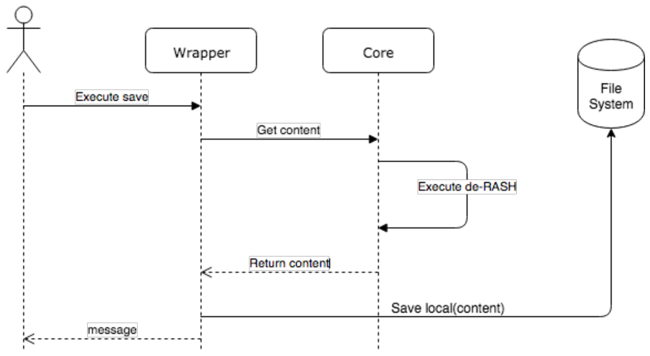


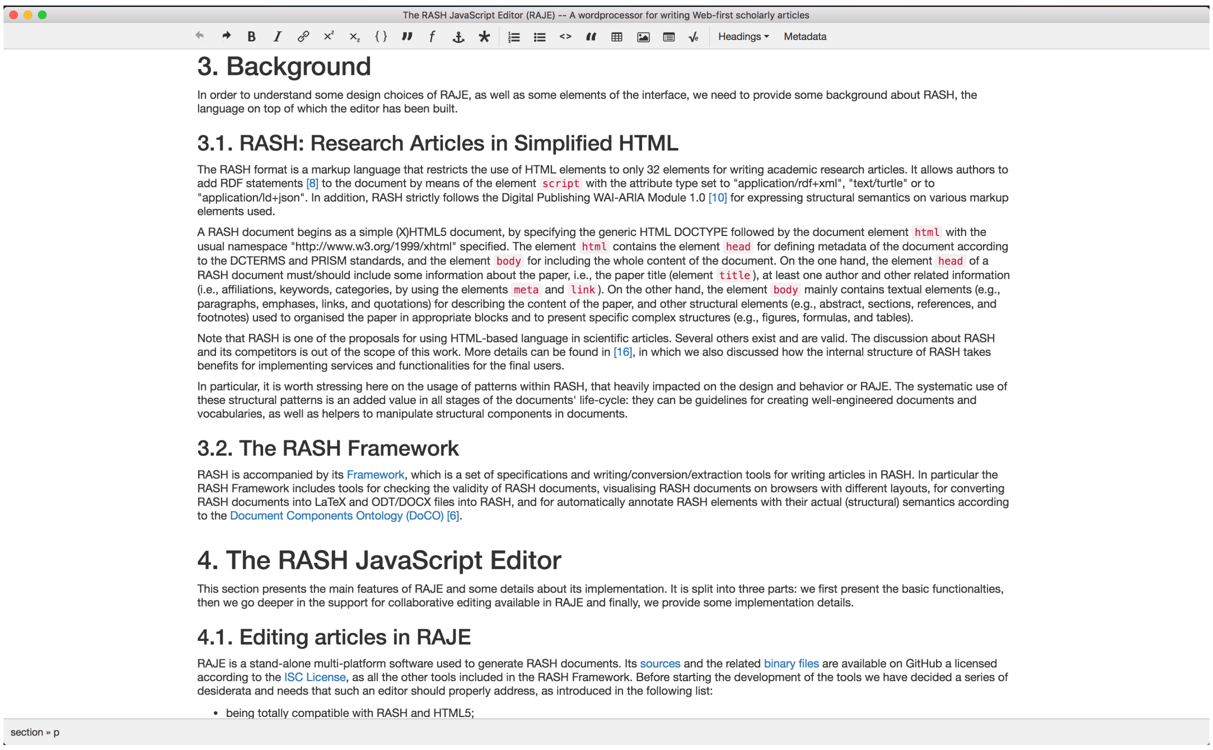
Figure 3: A UML diagram that shows how the save process implemented by RAJE-app (i.e. the wrapper) works.

easier versions management, and it also opens up to the development of further complex actions (to be implemented). Examples are merging shared articles and restoring a particular version of an article; note also that the whole set of Git functionalities is supported by NodeGit. It is worth mentioning that, while RAJE-app is already integrated with Git, the integration with GitHub is still under development.

3.3 Testing the wordprocessor

The new version of the wordprocessor, i.e. RAJE-app, is depicted in Fig. 4. It is a quite simple and intuitive interface, composed by a toolbar, which allows one to add all the elements introduced in RASH, and the content of the article one is writing. This demo paper has been written by using RAJE-app, and it is available at `https://w3id.org/people/essepuntato/papers/raje-www2018.html`.

In the past months, we have also tested extensively the new version of the editor by using it to create the entire set of lecture notes of the Computational Thinking and Program-


The RASH JavaScript Editor (RAJE) -- A wordprocessor for writing Web-first scholarly articles
Headings
Metadata

3. Background

In order to understand some design choices of RAJE, as well as some elements of the interface, we need to provide some background about RASH, the language on top of which the editor has been built.

3.1. RASH: Research Articles in Simplified HTML

The RASH format is a markup language that restricts the use of HTML elements to only 32 elements for writing academic research articles. It allows authors to add RDF statements [8] to the document by means of the element `script` with the attribute type set to "application/rdf+xml", "text/turtle" or to "application/ld+json". In addition, RASH strictly follows the Digital Publishing WAI-ARIA Module 1.0 [10] for expressing structural semantics on various markup elements used.

A RASH document begins as a simple (X)HTML5 document, by specifying the generic HTML DOCTYPE followed by the document element `html` with the usual namespace "http://www.w3.org/1999/xhtml" specified. The element `html` contains the element `head` for defining metadata of the document according to the DCTERMS and PRISM standards, and the element `body` for including the whole content of the document. On the one hand, the element `head` of a RASH document must/should include some information about the paper, i.e., the paper title (element `title`), at least one author and other related information (i.e., affiliations, keywords, categories, by using the elements `meta` and `link`). On the other hand, the element `body` mainly contains textual elements (e.g., paragraphs, emphases, links, and quotations) for describing the content of the paper, and other structural elements (e.g., abstract, sections, references, and footnotes) used to organised the paper in appropriate blocks and to present specific complex structures (e.g., figures, formulas, and tables).

Note that RASH is one of the proposals for using HTML-based language in scientific articles. Several others exist and are valid. The discussion about RASH and its competitors is out of the scope of this work. More details can be found in [16], in which we also discussed how the internal structure of RASH takes benefits for implementing services and functionalities for the final users.

In particular, it is worth stressing here on the usage of patterns within RASH, that heavily impacted on the design and behavior or RAJE. The systematic use of these structural patterns is an added value in all stages of the documents' life-cycle: they can be guidelines for creating well-engineered documents and vocabularies, as well as helpers to manipulate structural components in documents.

3.2. The RASH Framework

RASH is accompanied by its Framework, which is a set of specifications and writing/conversion/extraction tools for writing articles in RASH. In particular the RASH Framework includes tools for checking the validity of RASH documents, visualising RASH documents on browsers with different layouts, for converting RASH documents into LaTeX and ODT/DOCX files into RASH, and for automatically annotate RASH elements with their actual (structural) semantics according to the Document Components Ontology (DoCO) [6].

4. The RASH JavaScript Editor

This section presents the main features of RAJE and some details about its implementation. It is split into three parts: we first present the basic functionalties, then we go deeper in the support for collaborative editing available in RAJE and finally, we provide some implementation details.

4.1. Editing articles in RAJE

RAJE is a stand-alone multi-platform software used to generate RASH documents. Its sources and the related binary files are available on GitHub a licensed according to the ISC License, as all the other tools included in the RASH Framework. Before starting the development of the tools we have decided a series of desiderata and needs that such an editor should properly address, as introduced in the following list:

- being totally compatible with RASH and HTML5;

section » p

Figure 4: A screenshot of RAJE-app, showing the RASH version of [4].

ming course[14] of the Digital Humanities and Digital Knowledge second-cycle degree at the University of Bologna, Italy. In practice, after the first stable version of the new RAJE-app was released in early November, one of the authors of this paper started to use it on daily basis for writing the lecture notes of the course. The official GitHub repository of RAJE-app was used to collect issues. This has brought to the creation of 12 releases[15] so far (as of 7 January 2018), which every time improved the stability of the editor and the functionalities it provides.

4. CONCLUSIONS

In this paper, we have briefly introduced the re-engineered version of RAJE, i.e. a wordprocessor for writing scholarly papers in HTML compliant with RASH. In particular, we have presented the new modules developed for implementing the tool, i.e. RAJE-core, which implements the editing features, and RAJE-app, i.e. the standalone wordprocessor that reuses RAJE-core and extends it with storing capabilities. We have also briefly described the main differences with the previous version of RAJE described in [4] in terms of functionalities, modularity and stability of the editor.

In the future, we plan to develop additional functionalities. For instance, we currently developing a mechanism for reading and writing annotations compliant with the Web Annotation Vocabulary [3], and we also plan to include a spell-checker in the editor so as to facilitate users in correcting grammar mistakes and typos.

[14]`https://github.com/essepuntato/comp-think/`
[15]`https://github.com/rash-framework/raje-app/releases`

Acknowledgements

We would like to thank all the participants in the user testing sessions described in [4], who have provided us useful feedback that has been reused for improving RAJE – and that resulted in the re-enginering process described in this paper.

5. REFERENCES

[1] Capadisli, S., Guy, A., Verborgh, R., Lange, C., Auer, S., Berners-Lee, T. (2017). Decentralised Authoring, Annotations and Notifications for a Read-Write-Web with dokieli. In Proceedings of the 17th International Conference on Web Engineering (ICWE 2017): 469-481. DOI: `https://doi.org/10.1007/978-3-319-60131-1_33`

[2] Peroni S., Osborne F., Di Iorio A., Nuzzolese A. G., Poggi F., Vitali F., Motta E. (2017). Research Articles in Simplified HTML: a Web-first format for HTML-based scholarly articles. PeerJ Computer Science, 3: e132. DOI: `https://doi.org/10.7717/peerj-cs.132`

[3] Sanderson, R., Ciccarese, P., Young, B. (2017). Web Annotation Vocabulary. W3C Recommendation 23 February 2017. `https://www.w3.org/TR/annotation-vocab/` (last visited 7 January 2018)

[4] Spinaci, G., Peroni, S., Di Iorio, A., Poggi, F., Vitali, F. (2017). The RASH JavaScript Editor (RAJE): A Wordprocessor for Writing Web-first Scholarly Articles. In Proceedings of the 2017 ACM Symposium on Document Engineering (DocEng 2017): 85-94. DOI: `http://doi.acm.org/10.1145/3103010.3103018`