Bibes

An Open Source Live Search of Publications in Javascript

By

JORGE A. RAMOS MOUKEL

Department of Electrical and Computer Engineering
UNIVERSITY OF NEW MEXICO

A Masters' project submitted in accordance with the requirements of the MASTERS degree in the Faculty of Engineering.

NOVEMBER 2016

CHAPTER 1

INTRODUCTION

The need of exposing publications (papers, books, articles, etc) to the public is very common among universities, labs, book publishers, and others, as a way for these organizations to demonstrate what they are capable of. This project, called Bibes, attempts to satisfy that need.

In short, Bibes is a program that converts a BibTeX file into a webpage. Several tools out there already provide a way of doing this, but they lack common browsing tools like search engines and filters. They also lack an atractive user interface. Some of them are slow, too old, and not responsive enough for modern devices like tablets and phones.

Tipically, out-of-date BibTeX files might contain publications that could potentially point to broken links for their PDFs, webpages, journals, etc. None of the other tools found offers a way to identify these broken links, which this project attempts to do.

The following chapters will explain how Bibes works. However, Bibes is already available online at *www.bibesjs.org*, so feel free to try the *demo* even before start reading the next pages.

CHAPTER 2

BIBES

2.1 Related Work

Tere are several tools out there capable of converting BibTeX files into HTML code (webpages). The next 4 are some of the most popular.

Bibtexbrowser is a PHP script that receives a BibTeX file and generates HTML code. When renderer in the browser, it allows the end user to filter the list of publications by year, type and authors and have a search engine. It's been used for more than 100 universities.

Bib2xhtml is a program that converts BibTeX files into HTML code (specifically XHTML 1.0). This is a fork of the bib2html program written by David Hull in 1996 and maintained until 1998.

Bibtex2html is a program that converts BibTeX files into HTML code. It allows the webmaster to sort the list of publications in the HTML code by dates or authors, and in increasing or decreasing order.

Finally, **Bibhtml** is a collection of BibTeX style files, which allow you to use a BibTeX file to produce bibliographies in HTML.

The webpages produced by these tools look like if they were built in the mid 90's. Bibes' user interface is better looking than the ones these tools show. None of these tools provides a reponsive (adaptable to many screen sizes) webpage like Bibes does. And some of these, like Bibtexbrowser, are very slow (more than 10 seconds to load). Only one have a search engine to search through the publications. And, unlike Bibes, none of them allows you to debug the broken links some

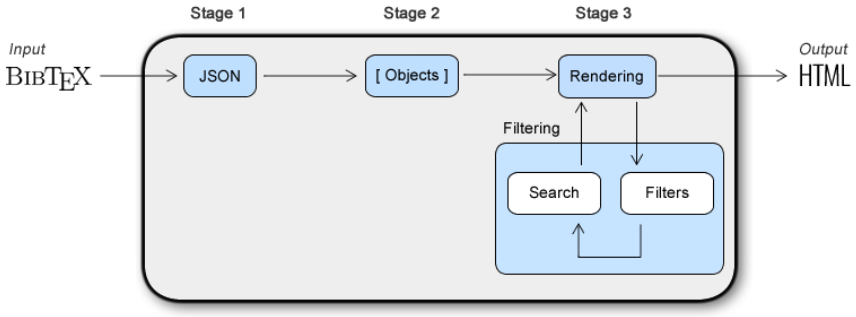

FIGURE 2.1. Bibe's Pipeline

publications might point to.

2.2 Design

The design of Bibes is very simple: it's a pipeline that receives a BibTex file as input, and produces HTML code (a webpage) as output. This pipelines is composed of 3 stages (see figure 2.1):

- Stage 1: Converting BibTeX to JSON
- Stage 2: Parsing JSON into Objects
- Stage 3: Rendering List of Publications

The code in these 3 stages was written in Javascript, CSS and HTML. The next 3 chapters explain each stage in detail.

2.3 Stage 1: Converting BibTeX to JSON

This stage converts the input BibTeX file into a string with a JSON format (see figure 2.2). Together with XML, the JSON format is one of the most common formats to save structured information in a string.

To convert the input BibTeX file (that contains the list of publications) into a JSON, BibtexParser was used. BibtexParser is a Python third-party library that traduces the contents of BibTex files into a JSON string (i.e. a string in JSON format). This stage can be found in method *startReadingBibtexFile( )* (see listing A.1 in line 256).

The main advantage of BibtexParser is how fast it executes the conversion of large BibTeX files.

FIGURE 2.2. Stage 1: Converting BibTeX to JSON

2.4 Stage 2: Objects

This stage receives as input the JSON string produced in the first stage. Given this JSON string contains the list of all the publications found in the input BibTeX file, the main goal of this stage is to convert (or parse) the JSON string into an array of objects (see figure 2.3) by using Javascript.

Each object in this output array is an instance of the Publication class which holds all the information of a publication (i.e. authors, title, description, etc).

This stage can be found in method *parseBibTexToJSON(...)* (see listing A.1 in line 267).

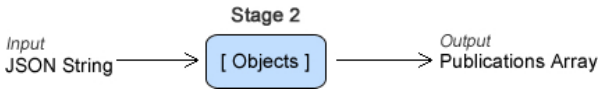

FIGURE 2.3. Stage 2: Creating array of publications

2.5 Stage 3: Rendering

This is the final stage and it's the one that contains most of the code. It is responsible for displaying the list of publications to the user in the browser. It also displays the ***filters*** and the ***search box*** (see figure 2.4).

This stage can be found in method *renderPublications( )* (see listing A.1 in line 422).

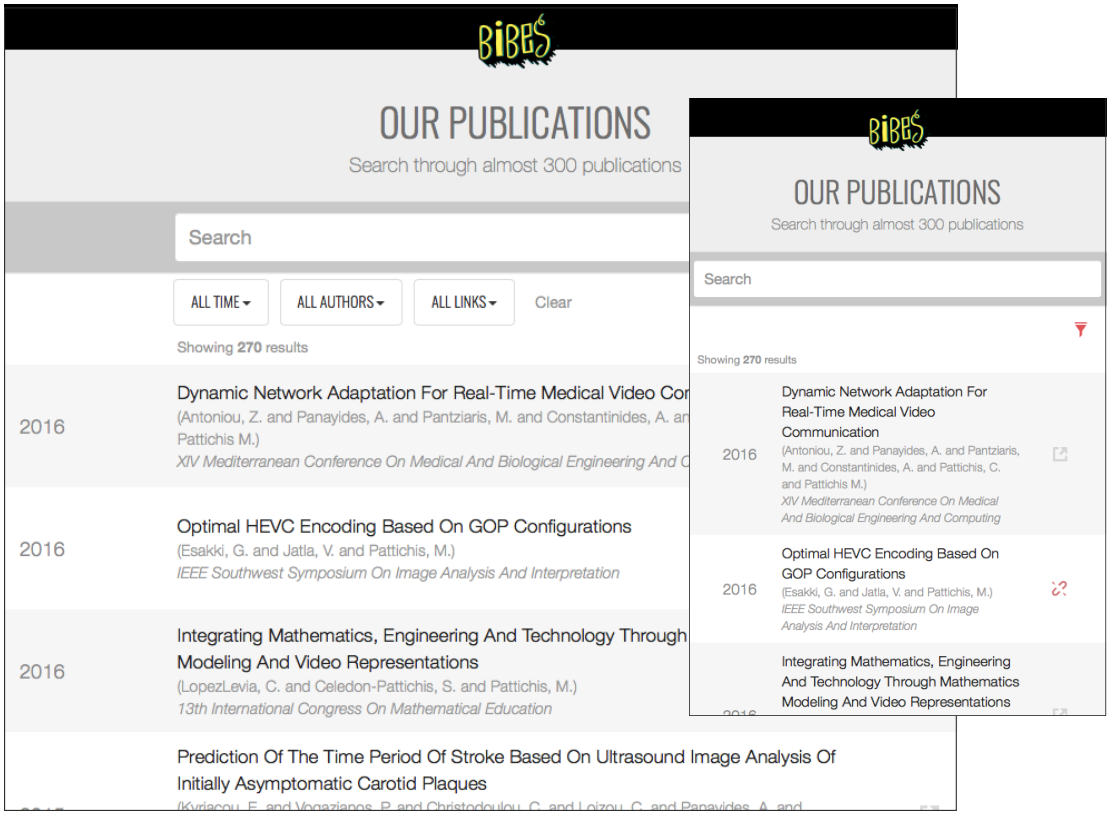

FIGURE 2.5. On the left, a screenshot of Bibes on a laptop. On the right, a screenshot of Bibes from a phone. To save some space, the filters buttons are hidden by default on the phone.

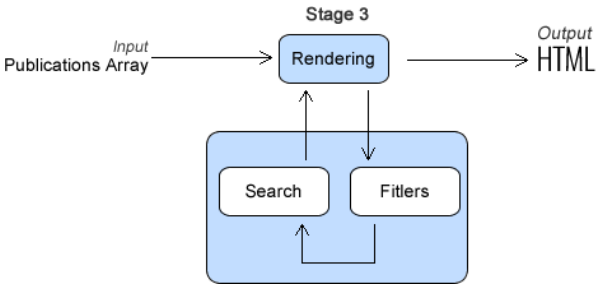

FIGURE 2.4. Stage 3: Rendering

2.5.1 Bootstrap

To render all the elements the end user will see in the webpage, the Bootstrap framework was used. Bootstrap allows a webpage to look good on any kind of device the user might be using. This means that Bibes is adjusted, with the proper presentation, whether the user is using a laptop, a phone, a tablet or any other device (see figure 2.5).

2.5.2 Filters

Bibes provides the user with the ability to filter the list of publications. There are 4 filters the user can use: ***by publication year***, ***by authors***, ***by working links*** and ***sort by***.

Every time the user sets one of the filters, the rendering process is repeated and Bibes iterates through all the publications, comparing each publication against all the filters set by the user. If the publication satisfies the filters (and the text in the search box), the publication is displayed to the user. Otherwise, the publication is not displayed.

The options found inside the filters ***by publication year*** and ***by authors*** are dinamically set. This means their options are set based on the publications found in input BibTeX file. In the case of the ***by publication year*** filter, the three most recent years (found in the list of publications) are offered as options to the user. In the case of the ***by author*** filter, the top 7 authors (the ones with the most publications) are displayed as options for this filter.

The ***by working links*** filter offers the user a way to show only publications with working links (i.e. links that are still valid and not broken). Also, this filter allows the opposite: to show only publications with broken links. The latter provides the owner of the input BibTeX file, a way to see the publications that contain broken links so the owner can fix them. This checking of broken links is made by Bibes in real time once the user loads the website. Which gives the user the most recent status of the links.

The ***sort by*** filter allows the user to sort publications *by most recent* and *oldest*.

2.5.3 Live Search

The live search works like a tipical search engine. It allows the user to type text and see only the publications that match that text. It's *live* because it displays the results of the search inmmediately as the user types each character in the search box. This live feature works asynchronous, so the user interface isn't blocked as the user types.

Bibe's live search uses regular expressions to match the text in the search box with the text found in four of the properties of each publication: *publication year*, *authors*, *title* and *description*. If one of these four properties contains the text in the search box, the associated publication (if it also matches the filters previously mentioned) will be displayed to the user. Otherwise, it won't be displayed.

2.5.4 Summary

Every time the user sets a filter or writes in the search box, the rendering process starts. Right before starting the rendering process, Bibes iterates through all the publications and compare each publication against all the filters set and the text in the search box. If the publication satisfies the filters and matches the text in the search box, the publication is displayed. Otherwise, the publication is not displayed.

2.6 Open Source

Bibes is an open source project under the MIT license. Which means that anyone has permition to use Bibes free of charge and without any restriction.

Bibes is currently available to the public on the website *www.bibesjs.org*. It can be downloaded from there and a simple 4-steps tutorial is offered to learn how to use it. It's also possible to easily customize Bibes, so instructions about customization are offered as well in the website.

Bibes is also available as repository on GitHub in case anyone wants to collaborate with it.

2.7 Future Work

Numerous new features could be implemented for Bibes, but for time constraints they could not be implemented on this project. The following are some of them.

2.7.1 Pagination

Bibes currently displays all the results in one single page, making it harder for the user to scroll down over a large number of publications. It would be better if Bibes could group the results in small pages. The first N results could be displayed in the first page, the second N results in the second page, and so on, just like modern search engines do.

To implement this, Bibes could simply make use of the pagination component provided by the Bootstrap framework. To do this, the function *renderPublications()* (see listing A.1 in line 422) in *bibes.js* file should be modify to render an N number of publications on each page page (instead of rendering them all in one).

2.7.2 Number of Citations

It'd be useful if each publication could show to the user the number of citations it has. It'd be also useful to sort the list of publications by the number of citations so the user can see the most

relevant publications first. The problem is that bibtex files do not come with this information, and if they did, it's data that needs to be updated often.

One way to achieve this would be using a third party API that provides the number of citations for a given publication ID (i.e. ISSN). The requests to this API could be done right after the function call *startReadingBibtexFile(parseBibTexToJSON)* (see listing A.1) is made (which is responsible for parsing the BibTeX file into objects).

2.7.3 Filter by Type

Some users might be interested in looking at only publications of certain type (like journals). That's why it could be useful to filter the publications by type.

To achieve this, a *type* filter should be created in the *template.html* file. Also, a handler should be created in *bibes.js* file to make the new filter respond to user interaction. Finally, method *getPublicationsListFromFilters()* (see listing A.1 in 530) should updated to filter the publication using the new filter criteria.

2.7.4 Favorites

The user might want to tag certain publications as his/her favorites in order to not lose track of those publications while browsing for others. Placing a *favorite* button next to each publication would be a solution for this. Also, a way to allow the user to only visualize favorite publications should also be provided (like a new filter).

2.7.5 Quick View

What if the user could see more detailed information for a given publication without having to open the link of that publication?

This could be implemented by showing to the user a small pop up window (when the user clicks on a publication) that shows a little more information about the publication. For this purpose, the BibTeX file should have enough extra information that could be presented to the user.

CHAPTER 3

CONCLUSION

Bibes presents itself as a modern alternative for those willing to have a fast, attractive and easy-to-setup live search of publications for their website. It's a bit surprising no other tool out there was found having these basic characteristics.

Bibes also presents itself as a debugging tool, by using the *by working links* filter, for those willing to fix their out-of-date BibTex files and avoid a bad user experience when clicking on those broken links.

Compared to other tools found, Bibes shows up as a modern system. Mainly because it adapts its user interface to modern devices like phones, tablets and laptops. And also because it offers both a live search and a set of filters that allow the end user to conveniently browse through all the publications.

APPENDIX A

A.1 References

[1] David Herman. 2013: Effective Javascript, 68 Specific Ways to Harness the Power of Javascript. Addison-Wesley.
[2] Michael Mahemoff. 2006: Ajax Design Patterns. O'Reilly.
[3] Francois Boulogne and other contributors (2015) BibtexParser's documentation!. Accessed 08/29/16. URL: *https://bibtexparser.readthedocs.io/en/v0.6.2/*
[4] Martin Monperrus (2007) bibtexbrowser: publication lists with bibtex and PHP. Accessed 09/6/16. URL: *http://www.monperrus.net/martin/bibtexbrowser/*
[5] D. Spinellis (2002-2010) bib2xhtml - Convert BibTeX Files into HTML. Accessed 09/7/16. URL: *http://www.spinellis.gr/sw/textproc/bib2xhtml/*
[6] Jean-Christophe.Filliatre. The bibtex2html home page. Accessed 09/7/16. URL: *https://www.lri.fr/filliatr/bibtex2html/*
[7] Norman Gray (2013). Bibhtml. Accessed 09/8/16. URL: *http://nxg.me.uk/dist/bibhtml/*
[7] Twitter (2011-2016). Bootstrap: The world's most popular mobile-first and responsive front-end framework. Accessed 09/10/16. URL: *http://getbootstrap.com/*
[8] GitHub (2011-2016). How people build software. Accessed 10/14/16. URL: *https://github.com/*
[9] Refsnes Data (1999-2016). W3School Online Web Tutorials Accessed 09/02/16. URL: *http://www.w3schools.com/*

A.2 How to Install Bibes

After downloading Bibes, follow the 4 steps in figure A.1:

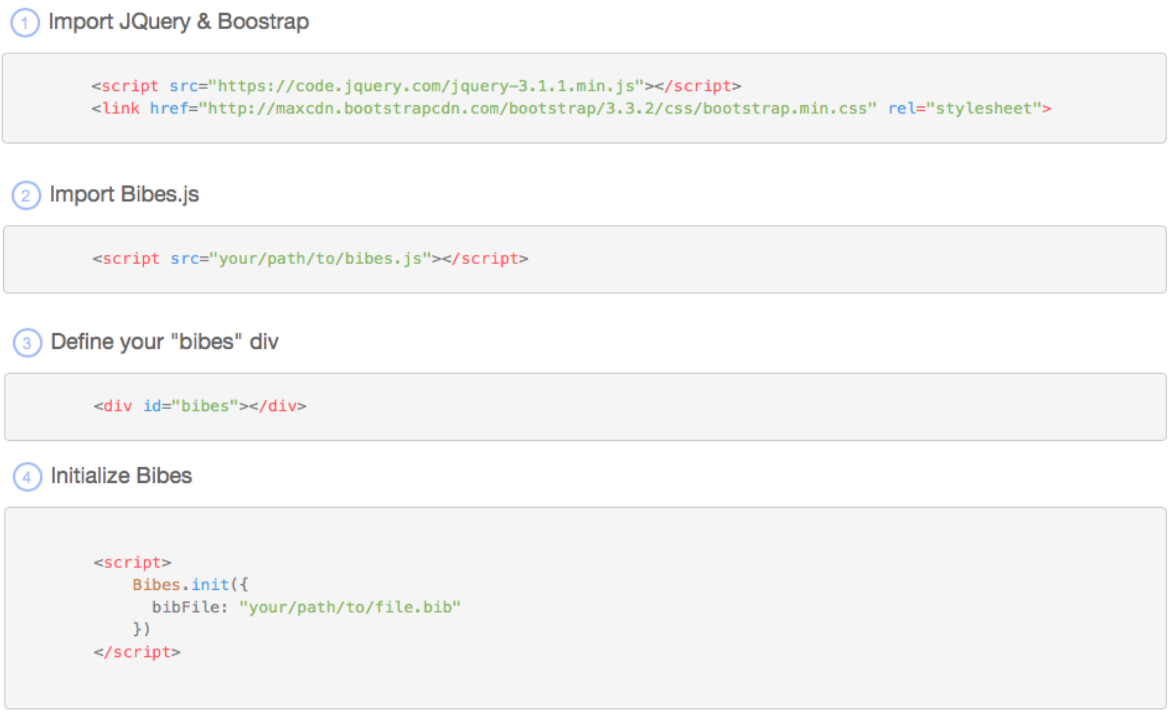

FIGURE A.1. The 4 steps needed to install Bibes

After following those 4 steps, your html code should end up looking like in figure A.2:

```
<head>
    ...
    <script src="https://code.jquery.com/jquery-3.1.1.min.js"></script>
    <link href="http://maxcdn.bootstrapcdn.com/bootstrap/3.3.2/css/bootstrap.min.css" rel="stylesheet">
    <script src="your/path/to/bibes.js"></script>
</head>

<body>
    ...
    <div id="bibes"></div>
    ...
    <script>
        Bibes.init({
          bibFile: "your/path/to/file.bib"
        })
    </script>
</body>
```

FIGURE A.2. How your code should end up looking

A.3 How to Customize Bibes

When initializing Bibes, just follow the example in figure A.3:

```
<script>
    Bibes.init({
      bibFile: "your/path/to/file.bib",
      data: {
        title: "My Title",
        subtitle: "My Subtitle",
        title_background_color: "#00FF00",
        search_bar_background_color: "blue"
      }
    })
</script>
```

FIGURE A.3. How to customize Bibes

As seen in figure A.3, you can use regular CSS colors (in hexadecimal, by name, etc) to set the background colors.

You can go much further with the customization by editing two files found in your Bibes' root folder. You can edit the styles by editing the file *css/bibes.css* and even edit the whole layout of the page by editing the file *index.html*.

A.4 Bibes' Javascript Code

The following listing shows the Javascript code of Bibes:

```
/*
The MIT License (MIT)
Copyright (c) <year> <copyright holders>

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and
    associated documentation files (the "Software"), to deal in the Software without restriction,
    including without limitation the rights to use, copy, modify, merge, publish, distribute,
    sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is
    furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial
     portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT
    NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
    NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM,
    DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
    OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
*/
(function (exports)
{
  "use strict";

  /// CONSTANTS
  var MAIN_DIV_CLASS_NAME              = "#bibes";
```

```
17  var NOT_FOUND_MAIN_DIV_ERROR_MESSAGE = "Bibes.js: 'bibes' div not found. Add a div with id 'bibes'
        to your webpage.";
18  var SECOND_MAIN_DIV_CLASS_NAME       = ".ivpcl-main-container";
19  var LOADING_SPINNER_CLASS_NAME       = ".ivpcl-loading-spinner";
20  var JUMBOTRON_ID_NAME                = "#ivpcl-jumbotron";
21  var SEARCH_DIV_CLASS_NAME            = ".ivpcl-publications-panel";
22  var TEMPLATE_FILE_URL                = "template/template.html";
23  var TITLE_DIV_CLASS_NAME             = ".ivpcl-header-title";
24  var SUBTITLE_DIV_CLASS_NAME          = ".ivpcl-header-subtitle";
25  var PUBLICATIONS_DIV_CLASS_NAME      = "publicationsDiv";
26  var TIME_FILTER_SELECTION_CLASS_ID   = "#time-filter-selection"
27  var TIME_FILTER_CLASS_ID             = "#time-filter"
28  var AUTHOR_FILTER_CLASS_ID           = "#authors-filter";
29  var AUTHOR_FILTER_SELECTION_CLASS_ID = "#authors-filter-selection";
30  var LINKS_FILTER_CLASS_ID            = "#links-filter";
31  var LINKS_FILTER_SELECTION_CLASS_ID  = "#links-filter-selection";
32  var SORTBY_FILTER_SELECTION_CLASS_ID = "#sortby-filter-selection";
33  var SORTBY_FILTER_CLASS_ID           = "#sortby-filter";
34  var CLEAR_BUTTON_CLASS_ID            = "#clear-button";
35  var RESULTS_TEXT_CLASS_ID            = "#results-number-text";
36  var SEARCH_TEXT_FIELD_CLASS_ID       = "#searchTextField";
37  var SEARCH_CLEAR_BUTTON_CLASS_ID     = "#searchClearButton";
38  var LOADING_SIGN_ID                  = ".ivcpl-loading-sign";
39  var ALL_AUTHORS_TEXT                 = "ALL AUTHORS";
40  var ALL_LINKS_TEXT                   = "ALL LINKS";
41  var ALL_TIME_TEXT                    = "ALL TIME";
42  var OLDER_TEXT                       = "OLDER";
43  var SORT_BY_TEXT                     = "SORT BY:";
44  var SORT_BY_MOST_RECENT_TEXT         = "MOST RECENT";
45  var SORT_BY_OLDEST_TEXT              = "OLDEST";
46  var LINKS_ONLY_WORKING               = "WORKING LINKS";
47  var LINKS_ONLY_NOT_WORKING           = "NO LINKS";
48  var LINKS_BROKEN                     = "BROKEN LINKS";
49  var DEFAULT_TITLE                    = "OUR PUBLICATIONS";
50  var DEFAULT_SUBTITLE_START           = "Search through"
51  var BIBTEX_PARSER_URL                = "packages/bibtex-parser-master/bibtexParse.js"
52  var NUMBER_OF_TIMES_IN_FILTER        = 3;
53  var NUMBER_OF_AUTHORS_IN_FILTER      = 7;
54  var INPUT_DATA_TITLE_NAME                          = "title";
55  var INPUT_DATA_SUBTITLE_NAME                       = "subtitle";
56  var INPUT_DATA_TITLE_BACKGROUND_COLOR_NAME         = "title_background_color";
57  var INPUT_DATA_SEARCH_BACKGROUND_COLOR_NAME        = "search_bar_background_color";
58  var WORKING_LINK_STATUS              = 0;
59  var MISSING_LINK_STATUS              = 1;
60  var BROKEN_LINK_STATUS               = 2;
61
62  /// GLOBAL VARIABLES
63  //
64  /// Content of the bibtex file but in JSON format.
65  var bibFileURL;
66  var bibTextJSON;
67  var publicationsArray;
68  var yearsArray, typesArray, authorsArray;
69  var selectedAuthor, selectedYear, selectedSortBy;
70  var renderPublicationsTimer;
71  var title, subtitle, titleBackgroundColor, searchBackgroundColor;
72
73  /*************************
74   *         OBJECTS       *
75   *************************/
76
77   /// Object that will contains the information of one publication.
78   function Publication(issn, author, title, booktitle, month, pages, realTitle, year, journal, url,
         volume, number, editor, address, note)
79   {
```

```
80      this.ISSN        = issn;
81      this.author      = author;
82      this.pubTitle    = title;
83      this.booktitle   = booktitle;
84      this.month       = month;
85      this.pages       = pages;
86      this.realTitle   = realTitle;
87      this.year        = year;
88      this.journal     = journal;
89      this.url         = url;
90      this.volume      = volume;
91      this.number      = number;
92      this.editor      = editor;
93      this.address     = address;
94      this.note        = note;
95      this.linkStatus = (url.length !== 0) ? WORKING_LINK_STATUS : MISSING_LINK_STATUS;
96   }
97
98   /*************************
99    *    PUBLIC FUNCTIONS    *
100   *************************/
101
102  /**
103   * Initializes the publications page.
104   *
105   */
106   exports.init = function({bibFile, data}={})
107   {
108      if ($(MAIN_DIV_CLASS_NAME).length === 0)
109        console.error(NOT_FOUND_MAIN_DIV_ERROR_MESSAGE);
110      else
111      {
112          // init vars
113          bibTextJSON       = "";
114          publicationsArray = [];
115          yearsArray        = [];
116          typesArray        = [];
117          authorsArray      = [];
118          bibFileURL        = bibFile;
119
120          // Inserting HTML code (from template) in the webpage that's initializing Bibes.JS.
121          insertTemplateHTMLCode();
122
123          /**********************************************************************
124           *  STAGE 1 & STAGE 2: From Bibtex to JSON and From JSON to Object.  *
125           **********************************************************************/
126
127          // Extracting content from Bibtext file.
128          startReadingBibtexFile(parseBibTexToJSON);
129
130          // Start
131          setTimeout(function ()
132          {
133            // Hidding loading spinner
134            $(LOADING_SPINNER_CLASS_NAME).hide();
135
136            // Displaying html content.
137            $(SECOND_MAIN_DIV_CLASS_NAME).fadeToggle( "slow", "linear" );
138
139            // Setting customization data set by the user into the HTML code.
140            setCustomizationInputDataToHTML(data);
141
142            /*********************
143            *  STAGE 3: Render   *
144            *********************/
```

```
145
146             // Rendering everything
147             renderAll();
148
149             // Setting initial filter values.
150             setFilterDefaults();
151
152             // Checking for broken links.
153             checkPublicationsWithBorkenLinks();
154
155         }, 500);
156     }
157 };
158
159 /*************************
160  *    PRIVATE FUNCTIONS   *
161  *************************/
162
163  /**
164   * It inserts the HTML code from the template in the webpage that's initializing Bibes.JS.
165   *
166   */
167  function insertTemplateHTMLCode()
168  {
169     // Loading HTML template into webpage.
170     $.ajax(
171     {
172        type: "GET",
173        url : TEMPLATE_FILE_URL,
174        async:false,
175        success: function(response)
176        {
177           // Extracting HTML
178           var doc = new DOMParser().parseFromString(response, 'text/html');
179
180           // Inserting html into the loading webpage.
181           $("head").append(doc.head.innerHTML);
182           $(MAIN_DIV_CLASS_NAME).html(doc.body.innerHTML);
183        }
184     });
185
186     // Loading Bibtex parser javascript.
187     $.ajax({
188       url: BIBTEX_PARSER_URL,
189       dataType: "script",
190       async:false
191     });
192  }
193
194  /**
195   * It checks if the links of all the publications are working properly or not.
196   *
197   */
198  function checkPublicationsWithBorkenLinks()
199  {
200      // Starting loading sign
201      $(LOADING_SIGN_ID).show();
202
203      for (var i=0; i<publicationsArray.length; i++)
204         urlExists(publicationsArray[i]);
205  }
206
207 /**
208  * It requests to get the webpage found in input "publication"'s URL.
209  *
```

```
210     * @param publication   Publication where the url will be gotten from.
211     */
212     function urlExists(publication)
213     {
214        if (typeof(publication) !== 'undefined' &&
215            publication.linkStatus !== MISSING_LINK_STATUS)
216        {
217            $.ajax(
218            {
219                url:      publication.url,
220                dataType: 'JSONP',
221                type:     'GET',
222                jsonpCallback: 'callback',
223                complete:  function(xhr)
224                {
225                    var requestStatus = xhr.status;
226
227                    if(requestStatus === 404 ||
228                       requestStatus === 500)
229                       // 404 not found
230                       publication.linkStatus = BROKEN_LINK_STATUS;
231                    else
232                       // 404 not found
233                       publication.linkStatus = WORKING_LINK_STATUS;
234
235                    // If this publication is the last one in the array of publications, we render the
                          publications again so the changes of these requests (if any) can be seen by
                        the user.
236                    if (publicationsArray.length > 0)
237                    {
238                      if (publication.pubTitle === publicationsArray[Math.round(publicationsArray.
                            length * 0.4)].pubTitle ||
239                          publication.pubTitle === publicationsArray[publicationsArray.length - 1].
                                pubTitle)
240                          renderPublications();
241
242                      // Hidding loading sign.
243                      if (publication.pubTitle === publicationsArray[publicationsArray.length - 1].
                            pubTitle)
244                         $(LOADING_SIGN_ID).hide();
245                    }
246                }
247            });
248       }
249    }
250
251    /**
252     * It starts reading the content of the bibtex file which path is in constant BIBTEXT_FILE_URL.
            When complete, it calls the input function "completionFunction" to handle the content read.
253     *
254     * @param completionFunction    Function to be called when the reading complete. This function is
            supposed to receive a string with the content read from the bibtex file.
255     */
256     function startReadingBibtexFile(completionFunction)
257     {
258      // Extracting bibtex file content.
259        $.get(bibFileURL, completionFunction);
260     }
261
262    /**
263     * It parses the input "data" and converts it to JSON.
264     *
265     * @param data    String extracted from a BibText file.
266     */
267     function parseBibTexToJSON(data)
```

```
268   {
269     // Converting bibtext content string to JSON.
270     bibTextJSON = bibtexParse.toJSON(data);
271
272     // Extracting the info from the JSON object and putting it in the publications array.
273     for (var i=0; i<bibTextJSON.length; i++)
274     {
275       var issn, author, title, booktitle, month, pages, realTitle, year, journal, url, volume,
              number, editor, address, note;
276       var entryTag = bibTextJSON[i].entryTags;
277
278       // Extracting info from entry tag.
279       issn      = (typeof(entryTag.ISSN)     !== 'undefined') ? entryTag.ISSN       : "";
280       author    = (typeof(entryTag.author)    !== 'undefined') ? entryTag.author     : "";
281       title     = (typeof(entryTag.title)     !== 'undefined') ? entryTag.title      : "";
282       booktitle = (typeof(entryTag.booktitle) !== 'undefined') ? entryTag.booktitle  : "";
283       month     = (typeof(entryTag.month)     !== 'undefined') ? entryTag.month      : "";
284       pages     = (typeof(entryTag.pages)    !== 'undefined') ? entryTag.pages      : "";
285       realTitle = (typeof(entryTag.realTitle) !== 'undefined') ? entryTag.realTitle  : "";
286       year      = (typeof(entryTag.year)     !== 'undefined') ? entryTag.year       : "";
287       journal   = (typeof(entryTag.journal)   !== 'undefined') ? entryTag.journal    : "";
288       url       = (typeof(entryTag.url)    !== 'undefined') ? entryTag.url      : "";
289       volume    = (typeof(entryTag.volume)  !== 'undefined') ? entryTag.volume     : "";
290       number    = (typeof(entryTag.number)  !== 'undefined') ? entryTag.number     : "";
291       editor    = (typeof(entryTag.editor)  !== 'undefined') ? entryTag.editor     : "";
292       address   = (typeof(entryTag.address) !== 'undefined') ? entryTag.address    : "";
293       note      = (typeof(entryTag.note)    !== 'undefined') ? entryTag.note       : "";
294
295       // Checking the year is numberic
296       if (isNumeric(year) == false)
297         year = 0;
298
299       // Adding year (if not added yet) to the array of years.
300       insertYearInYearsArray(year);
301
302       // Adding author (if not added yet) to the array of authors.
303       insertAuthorsInAuthorsArray(author.split("and"));
304
305         // Creating a new pulication object with the extracted info.
306       var currentPublication = new Publication(issn, author, title, booktitle, month, pages,
              realTitle, year, journal, url, volume, number, editor, address, note);
307
308       // Adding publication to the array of publications
309       publicationsArray.push(currentPublication);
310     }
311
312     // Uncomment ONLY for debugging: printing the Bibtext content but in JSON format.
313     //console.log(bibTextJSON);
314     //console.log(publicationsArray);
315     //console.log(authorsArray);
316   }
317
318   /**
319    * Returns TRUE, if the input "n" is numeric. FALSE, otherwise.
320    *
321    * @param n    value which will be determined if it's numeric or not.
322    * @return TRUE, if the input "n" is numeric. FALSE, otherwise.
323    */
324   function isNumeric(n)
325   {
326      return !isNaN(parseFloat(n)) && isFinite(n);
327   }
328
329  /**
330   * Returns TRUE, if the input "x" is NaN. Returns FALSE, otherwise.
```

```
331   *
332   * @param x   Input that will be evaluated.
333   * @return TRUE, if the input "x" is NaN. Returns FALSE, otherwise.
334   */
335   function isNaN(x)
336   {
337    return x !== x;
338   }
339
340  /**
341   *  It sets the customization data received from the user, found in input "data",into the webpage
          HTML.
342   *
343   * @param data      Input received from the user.
344   */
345   function setCustomizationInputDataToHTML(data)
346   {
347       if ((typeof(data) !== 'undefined'))
348       {
349          // Extracting info from input "data".
350          title                 = data[INPUT_DATA_TITLE_NAME];
351          subtitle              = data[INPUT_DATA_SUBTITLE_NAME];
352          titleBackgroundColor  = data[INPUT_DATA_TITLE_BACKGROUND_COLOR_NAME];
353          searchBackgroundColor = data[INPUT_DATA_SEARCH_BACKGROUND_COLOR_NAME];
354
355          // Setting input into HTML code.
356          if ((typeof(title) !== 'undefined'))
357            $(TITLE_DIV_CLASS_NAME).html(title.toUpperCase());
358          else
359            $(TITLE_DIV_CLASS_NAME).html(DEFAULT_TITLE.toUpperCase());
360          if ((typeof(subtitle) !== 'undefined'))
361            $(SUBTITLE_DIV_CLASS_NAME).html(subtitle);
362          if ((typeof(titleBackgroundColor) !== 'undefined'))
363            $(JUMBOTRON_ID_NAME).css("background-color", titleBackgroundColor, 'important');
364          if ((typeof(searchBackgroundColor) !== 'undefined'))
365            $(SEARCH_DIV_CLASS_NAME).css("background-color", searchBackgroundColor);
366       }
367       else
368       {
369          var connectingWord = "";
370          var roundedNumberOfPublications = 0;
371          var firstDigit = 0, secondDigit = 0, firstTwoDigits;
372          var numberOfPublications = publicationsArray.length;
373
374          // Setting default values into HTML code.
375          firstDigit     = numberOfPublications % 10;
376          secondDigit    = Math.round(numberOfPublications / 10) % 10;
377          firstTwoDigits = secondDigit * 10 + firstDigit;
378
379          // Picking the subttitle to use based on the number of publications.
380          if (firstTwoDigits >= 50)
381          {
382            roundedNumberOfPublications = numberOfPublications + 100 - firstTwoDigits;
383            connectingWord = "almost";
384          }
385          else if (firstTwoDigits >= 25)
386          {
387            roundedNumberOfPublications = numberOfPublications + 50 - firstTwoDigits;
388            connectingWord = "almost";
389          }
390          else
391          {
392            roundedNumberOfPublications = numberOfPublications - firstTwoDigits;
393            connectingWord = "more than";
394          }
```

```
395
396             // Setting default title and subtitle.
397             $(TITLE_DIV_CLASS_NAME).html(DEFAULT_TITLE.toUpperCase());
398             $(SUBTITLE_DIV_CLASS_NAME).html(DEFAULT_SUBTITLE_START + " " + connectingWord + " " +
                    roundedNumberOfPublications + " publications");
399         }
400     }
401
402     /**
403      *  It renders all: filters and publications.
404      *
405      */
406     function renderAll()
407     {
408       // Sorting filters arrays.
409       sortFilterArrays();
410
411       // Rendering filters
412       renderFilters();
413
414       // Rendering publications
415       renderPublications();
416     }
417
418     /**
419      *  It renders the content extracted from the BibTex file.
420      *
421      */
422     function renderPublications()
423     {
424       if (isNaN(bibTextJSON) == false)
425       {
426         var htmlToRender = "";
427         var publications = getPublicationsListFromFilters();
428
429         // Setting number of results text.
430         $(RESULTS_TEXT_CLASS_ID).html("Showing <b>" + publications.length + "</b> results");
431
432         // Going through every publication.
433         for (var i=0; i<publications.length; i++)
434         {
435           var searchValue = $(SEARCH_TEXT_FIELD_CLASS_ID).val();
436           var regExp;
437
438           if (searchValue.length > 0)
439             regExp = new RegExp(searchValue + "+", "i");
440           else
441             regExp = new RegExp("");
442
443           // Starting Row
444           if (i % 2 !== 0)
445             htmlToRender += "<tr><td class='col-lg-1 col-md-1 col-sm-1 col-xm-1'></td>";
446           else
447             htmlToRender += "<tr style='background-color:#F6F6F6'><td class='col-lg-1 col-md-1 col-sm
                    -1 col-xm-1'></td>";
448
449           // Year
450           htmlToRender += "<td class='col-lg-2 col-md-2 col-sm-2 col-xm-2 text-center ivcpl-
                publication-year'>";
451           htmlToRender += publications[i].year.length !== 0 && publications[i].year != 0 ?
                publications[i].year.replace(regExp, "<b>" + searchValue + "</b>") : "-";
452           htmlToRender += "</td>";
453
454           // Description
```

```
455         htmlToRender += "<td class='col-lg-6 col-md-6 col-sm-6 col-xm-6 text-left ivcpl-publication-
                titleAuthorPlace'>";
456         htmlToRender += "<div class='ivcpl-publication-description-title'>";
457         htmlToRender += publications[i].pubTitle.length !== 0 ? publications[i].pubTitle.replace(
                regExp, "<b>" + searchValue + "</b>") : "Unknown Title";
458         htmlToRender += "</div>";
459         htmlToRender += "<div class='ivcpl-publication-description-subtitle'>";
460         htmlToRender += publications[i].author.length !== 0 ?  "(" + publications[i].author.replace(
                regExp, "<b>" + searchValue + "</b>") + ")" : "";
461         htmlToRender += "</div>";
462         htmlToRender += "<div class='ivcpl-publication-description-place'>";
463         htmlToRender += publications[i].booktitle.length !== 0 ? publications[i].booktitle.replace(
                regExp, "<b>" + searchValue + "</b>") : "";
464         htmlToRender += "</div>";
465         htmlToRender += "</td>";
466
467         // Downloads
468         htmlToRender += "<td class='col-lg-1 col-md-1 col-sm-1 col-xm-1 text-center ivcpl-
                publication-downloads'>";
469         if (publications[i].linkStatus === WORKING_LINK_STATUS)
470           htmlToRender += "<a target='_blank' href='"+ publications[i].url +"'><img src='pics/
                  urlIcon.png' title='Working link'></a>";
471         else if (publications[i].linkStatus === MISSING_LINK_STATUS)
472           htmlToRender += "<img src='pics/urlDisableIcon.png' title='Missing link'>";
473         else if (publications[i].linkStatus === BROKEN_LINK_STATUS)
474           htmlToRender += "<a target='_blank' href='"+ publications[i].url +"'><img src='pics/
                  brokenUrlIcon.png' width='26px' title='Broken link'></a>";
475
476         htmlToRender += "</td>";
477
478         // Ending row
479         htmlToRender += "<td class='col-lg-2 col-md-2 col-sm-2 col-xm-2'></td></tr>";
480       }
481
482       // Showing Empty List Message, if necessary.
483       if (publications.length == 0)
484         htmlToRender = "<tr><td class='col-lg-6 col-md-6 col-lg-offset-3 col-md-offset-3 ivcpl-
                publication-description-subtitle text-center'>No publications found</td></tr>";
485
486       // Inserting html into the div.
487       $("#" + PUBLICATIONS_DIV_CLASS_NAME).html(htmlToRender);
488     }
489
490     // Stopping timer
491     clearInterval(renderPublicationsTimer);
492   }
493
494   /**
495    * It renders the filters' options.
496    *
497    */
498   function renderFilters()
499   {
500       // Selecting the most recent years as options for the Time filter.
501       $(TIME_FILTER_CLASS_ID).html("");
502       $(TIME_FILTER_CLASS_ID).append("<li><a href='javascript:void(0);'>" + ALL_TIME_TEXT + "</a></
              li>");
503       for (var i=0; i<yearsArray.length && i<NUMBER_OF_TIMES_IN_FILTER; i++)
504         $(TIME_FILTER_CLASS_ID).append("<li><a href='javascript:void(0);'>" + yearsArray[i].
                toUpperCase() + "</a></li>");
505       $(TIME_FILTER_CLASS_ID).append("<li><a href='javascript:void(0);'>" + OLDER_TEXT + "</a></li>"
              );
506
507       // Selecting the first 5 authors with the most publications.
508       $(AUTHOR_FILTER_CLASS_ID).html("");
```

```
509       $(AUTHOR_FILTER_CLASS_ID).append("<li><a href='javascript:void(0);'>" + ALL_AUTHORS_TEXT + "</
              a></li>");
510       for (var i=0; i<authorsArray.length && i<NUMBER_OF_AUTHORS_IN_FILTER; i++)
511         $(AUTHOR_FILTER_CLASS_ID).append("<li><a href='javascript:void(0);'>" + authorsArray[i][0].
                toUpperCase() + " (" + authorsArray[i][1] + ")</a></li>");
512
513       // Selecting the first 5 authors with the most publications.
514       $(LINKS_FILTER_CLASS_ID).html("");
515       $(LINKS_FILTER_CLASS_ID).append("<li><a href='javascript:void(0);'>" + ALL_LINKS_TEXT + "</a
              ></li>");
516       $(LINKS_FILTER_CLASS_ID).append("<li><a href='javascript:void(0);'>" + LINKS_ONLY_WORKING + "
              </a></li>");
517       $(LINKS_FILTER_CLASS_ID).append("<li><a href='javascript:void(0);'><img class='ivcpl-loading-
              sign' src='pics/loading_spinner.gif' width='16px' title='Testing publications links...'>
               " + LINKS_BROKEN + "</a></li>");
518       $(LINKS_FILTER_CLASS_ID).append("<li><a href='javascript:void(0);'>" + LINKS_ONLY_NOT_WORKING
              + "</a></li>");
519
520       // Adding sorting typesArray
521       $(SORTBY_FILTER_CLASS_ID).html("");
522       $(SORTBY_FILTER_CLASS_ID).append("<li><a href='javascript:void(0);'>" +
              SORT_BY_MOST_RECENT_TEXT + "</a></li>");
523       $(SORTBY_FILTER_CLASS_ID).append("<li><a href='javascript:void(0);'>" + SORT_BY_OLDEST_TEXT +
              "</a></li>");
524   }
525
526   /**
527    * It returns the list of publications to be shown to the user taking into account the selections
           of each filter.
528    *
529    */
530   function getPublicationsListFromFilters()
531   {
532     var filteredPublicationsArray = [];
533
534     for (var i=0; i<publicationsArray.length; i++)
535     {
536       var publicationSufficesAllFilters = true;
537       var content;
538
539       // Checking the Time filter.
540       content = $(TIME_FILTER_SELECTION_CLASS_ID).html();
541       if (content.indexOf(ALL_TIME_TEXT) === -1)
542       {
543         // Cleaning content.
544         content = content.replace('TIME:', '');
545         content = content.replace('<span class="caret"></span>', '');
546         content = content.replace(/ /gi, '');
547         content = content.replace('&lt;', '< ');
548
549         if (content.localeCompare(OLDER_TEXT) === 0)
550         {
551           for (var k=0; k<yearsArray.length && k<NUMBER_OF_TIMES_IN_FILTER; k++)
552           {
553             if (publicationsArray[i].year == yearsArray[k])
554               publicationSufficesAllFilters = false;
555           }
556         }
557         else if ((publicationsArray[i].year + "").indexOf(content) === -1)
558           publicationSufficesAllFilters = false;
559       }
560
561       // Checking the Authors filter.
562       content = $(AUTHOR_FILTER_SELECTION_CLASS_ID).html();
563       if (content.indexOf(ALL_AUTHORS_TEXT) === -1 &&
```

```
564             publicationSufficesAllFilters)
565         {
566           // Cleaning content.
567           content = content.replace('AUTHOR:', '');
568           content = content.replace('<span class="caret"></span>', '');
569           content = content.replace('(', '');
570           content = content.replace(')', '');
571           content = content.replace('0', '');
572           content = content.replace('1', '');
573           content = content.replace('2', '');
574           content = content.replace('3', '');
575           content = content.replace('4', '');
576           content = content.replace('5', '');
577           content = content.replace('6', '');
578           content = content.replace('7', '');
579           content = content.replace('8', '');
580           content = content.replace('9', '');
581           content = content.replace(/ /gi, '');
582
583           if (publicationsArray[i].author.toUpperCase().indexOf(content) === -1)
584             publicationSufficesAllFilters = false;
585         }
586
587         // Checking the Links filter.
588         content = $(LINKS_FILTER_SELECTION_CLASS_ID).html();
589         if (content.indexOf(ALL_LINKS_TEXT) === -1 &&
590             publicationSufficesAllFilters)
591         {
592           if (content.indexOf(LINKS_BROKEN) !== -1)
593           {
594             // Selected option is: Borken links
595
596             if (publicationsArray[i].linkStatus !== BROKEN_LINK_STATUS)
597               publicationSufficesAllFilters = false;
598           }
599           else if (content.indexOf(LINKS_ONLY_NOT_WORKING) === -1)
600           {
601             // Selected option is: Working links
602
603             if (publicationsArray[i].linkStatus === MISSING_LINK_STATUS ||
604                 publicationsArray[i].linkStatus === BROKEN_LINK_STATUS)
605               publicationSufficesAllFilters = false;
606           }
607           else
608           {
609             // Selected option is: No links
610
611             if (publicationsArray[i].linkStatus === WORKING_LINK_STATUS ||
612                 publicationsArray[i].linkStatus === BROKEN_LINK_STATUS)
613               publicationSufficesAllFilters = false;
614           }
615         }
616
617         // Checking if publication suffices the search text.
618         if (publicationSufficesAllFilters)
619         {
620             var searchValue = $(SEARCH_TEXT_FIELD_CLASS_ID).val();
621
622             if (searchValue.length > 0)
623             {
624                 var regExp = new RegExp(searchValue + "+", "i");
625                 publicationSufficesAllFilters = regExp.test(publicationsArray[i].pubTitle) || regExp
                        .test(publicationsArray[i].booktitle) || regExp.test(publicationsArray[i].
                        author) || regExp.test(publicationsArray[i].year);
626             }
```

```
627         }
628
629         // Adding filtered publication to the array of publications to be shown to the user.
630         if (publicationSufficesAllFilters)
631           filteredPublicationsArray.push(publicationsArray[i]);
632       }
633
634       // Cleaning content.
635       content = $(SORTBY_FILTER_SELECTION_CLASS_ID).html();
636       content = content.replace('<span class="caret"></span>', '');
637       content = content.replace(/ /gi, '');
638
639       // Checking the SortBy filter.
640       if (content.length !== 0)
641       {
642           if (content.indexOf(SORT_BY_MOST_RECENT_TEXT) !== -1)
643           {
644              filteredPublicationsArray.sort(function(a, b) {return b.year - a.year});;
645           }
646           else if (content.indexOf(SORT_BY_OLDEST_TEXT) !== -1)
647           {
648              filteredPublicationsArray.sort(function(a, b) {return a.year - b.year});;
649           }
650       }
651
652       return filteredPublicationsArray;
653     }
654
655     /**
656      * It sorts all the arrays related to the filters.
657      *
658      */
659     function sortFilterArrays()
660     {
661        yearsArray.sort(function(a, b) {return b - a});
662        authorsArray.sort(function(a, b) {return b[1] - a[1]});
663
664        if (NUMBER_OF_TIMES_IN_FILTER <= yearsArray.length)
665          OLDER_TEXT = "< " + yearsArray[NUMBER_OF_TIMES_IN_FILTER - 1]
666     }
667
668       /**
669        * It sets the default values for all the filters.
670        *
671        */
672     function setFilterDefaults()
673     {
674        handleFilterSelection(ALL_TIME_TEXT, TIME_FILTER_SELECTION_CLASS_ID);
675        handleFilterSelection(ALL_AUTHORS_TEXT, AUTHOR_FILTER_SELECTION_CLASS_ID);
676        handleFilterSelection(ALL_LINKS_TEXT, LINKS_FILTER_SELECTION_CLASS_ID);
677        handleFilterSelection(SORT_BY_TEXT, SORTBY_FILTER_SELECTION_CLASS_ID);
678     }
679
680     /**
681      * It handles the selection given by input "newSelection" made by the user in the filter which id
            is given by input "filterClassId".
682      *
683      * @param newSelection    Selection made by the user in the Time Filter.
684      * @param filterClassId   ID of the class of the filter where the selection was made.
685      */
686     function setFilterSelection(newSelection, filterClassId)
687     {
688       if (newSelection.length !== 0)
689       {
690         $(filterClassId).empty();
```

```
691       $(filterClassId).append(newSelection + '<span class="caret"></span>');
692
693       // Printing for debugging
694       //console.log(newSelection);
695     }
696   }
697
698   /**
699    * It inserts the input year "year" in the array of years.
700    *
701    * @param year    Year to be inserted.
702    */
703   function insertYearInYearsArray(year)
704   {
705       if (year.length !== 0 &&
706           jQuery.inArray( year, yearsArray ) === -1)
707       yearsArray.push(year);
708   }
709
710   /**
711    * It inserts the authors found in input "sub_authorsArray" into the authors' array.
712    *
713    * @param year    Year to be inserted.
714    */
715   function insertAuthorsInAuthorsArray(sub_authorsArray)
716   {
717     for (var j=0; j<sub_authorsArray.length; j++)
718     {
719       if (sub_authorsArray[j].length !== 0)
720       {
721         var indexOfAuthor = -1;
722         var newAuthor = sub_authorsArray[j];
723
724         // Cleaning new author name.
725         newAuthor = newAuthor.split(",")[0];
726         newAuthor = newAuthor.replace(/ /gi, '');
727
728         // Checking if this author is already in the array
729         for (var i=0; i<authorsArray.length; i++)
730         {
731           if (authorsArray[i][0].localeCompare(newAuthor) == 0)
732           {
733             indexOfAuthor = i;
734             break;
735           }
736         }
737
738         // Inserting author
739         if (indexOfAuthor === -1)
740           authorsArray.push([newAuthor, 1]);
741         else
742           authorsArray[indexOfAuthor] = [newAuthor, authorsArray[indexOfAuthor][1] + 1];
743       }
744     }
745   }
746
747   /**
748    * It handles the selection given by input "selection" made by the user in the filter which id is
          given by input "filterClassId".
749    *
750    * @param selection       Selection made by the user in the Time Filter.
751    * @param filterClassId   ID of the class of the filter where the selection was made.
752    */
753   function handleFilterSelection(selection, filterClassId)
754   {
```

```
755     if (selection.length !== 0)
756     {
757       // Setting new selectiong in filter.
758       setFilterSelection(selection, filterClassId);
759
760       // Rendering publications again to reflect the selection.
761       renderPublications();
762     }
763    }
764
765   /**************************
766    *     EVENT HANDLERS     *
767    **************************/
768
769   /**
770    *  It handles the clicks in the search TIME filter.
771    */
772   $(function()
773   {
774       $(TIME_FILTER_CLASS_ID).bind('click', function (e) {
775         handleFilterSelection(e.target.innerHTML, TIME_FILTER_SELECTION_CLASS_ID);
776       });
777   });
778
779   /**
780    *  It handles the clicks in the search AUTHORS filter.
781    */
782   $(function()
783   {
784       $(AUTHOR_FILTER_CLASS_ID).bind('click', function (e)
785       {
786         if (e.target.innerHTML.localeCompare(ALL_AUTHORS_TEXT) == 0)
787           handleFilterSelection(e.target.innerHTML, AUTHOR_FILTER_SELECTION_CLASS_ID);
788         else
789           handleFilterSelection("AUTHOR: " + e.target.innerHTML, AUTHOR_FILTER_SELECTION_CLASS_ID);
790       });
791   });
792
793   /**
794    *  It handles the clicks in the search LINKS filter.
795    */
796   $(function()
797   {
798       $(LINKS_FILTER_CLASS_ID).bind('click', function (e)
799       {
800         if (e.target.innerHTML.localeCompare(ALL_LINKS_TEXT) == 0)
801           handleFilterSelection(e.target.innerHTML, LINKS_FILTER_SELECTION_CLASS_ID);
802         else
803           handleFilterSelection(e.target.innerHTML, LINKS_FILTER_SELECTION_CLASS_ID);
804       });
805   });
806
807   /**
808    *  It handles the clicks in the search SORTBY filter.
809    */
810   $(function()
811   {
812       $(SORTBY_FILTER_CLASS_ID).bind('click', function (e) {
813         handleFilterSelection(SORT_BY_TEXT + " " + e.target.innerHTML,
                 SORTBY_FILTER_SELECTION_CLASS_ID);
814       });
815   });
816
817   /**
818    *  It handles the clicks in the Clear button.
```

```
819     */
820   $(function()
821   {
822       $(CLEAR_BUTTON_CLASS_ID).bind('click', function (e) {
823           setFilterDefaults();
824       });
825   });
826
827   /**
828    *  It handles the typing in the Search text field.
829    */
830   $(function()
831   {
832       $(SEARCH_TEXT_FIELD_CLASS_ID).bind('keyup', function (e) {
833
834         // Rendering publications again to reflect the selection.
835         clearInterval(renderPublicationsTimer);
836         renderPublicationsTimer = setInterval(renderPublications, 150);
837
838         if ($(SEARCH_TEXT_FIELD_CLASS_ID).val().length > 0)
839           $(SEARCH_CLEAR_BUTTON_CLASS_ID).show()
840         else
841           $(SEARCH_CLEAR_BUTTON_CLASS_ID).hide()
842       });
843   });
844
845   /**
846    *  It handles the Clear Search button.
847    */
848   $(function()
849   {
850       $(SEARCH_CLEAR_BUTTON_CLASS_ID).bind('click', function (e)
851       {
852           $(SEARCH_TEXT_FIELD_CLASS_ID).val("");
853           $(SEARCH_CLEAR_BUTTON_CLASS_ID).hide()
854
855           // Rendering publications again to reflect the change.
856           renderPublications();
857       });
858   });
859 })
860 (typeof exports === 'undefined' ? this['Bibes'] = {} : exports);
861
862 \label{list:code}
```

Listing A.1: Bibes' Javascript Code