

Citation.js: a format-independent, modular bibliography tool for the browser and command line (#33959)

1

First submission

Editor guidance

Please submit by **30 Jan 2019** for the benefit of the authors (and your \$200 publishing discount).



Structure and Criteria

Please read the 'Structure and Criteria' page for general guidance.



Raw data check

Review the raw data. Download from the location [described by the author](#).



Image check

Check that figures and images have not been inappropriately manipulated.

Privacy reminder: If uploading an annotated PDF, remove identifiable information to remain anonymous.

Files

Download and review all files from the [materials page](#).

3 Figure file(s)

2 Latex file(s)

2 Raw data file(s)



Structure your review

The review form is divided into 5 sections. Please consider these when composing your review:

1. **BASIC REPORTING**
2. **EXPERIMENTAL DESIGN**
3. **VALIDITY OF THE FINDINGS**
4. General comments
5. Confidential notes to the editor






 You can also annotate this PDF and upload it as part of your review

When ready [submit online](#).





Editorial Criteria

Use these criteria points to structure your review. The full detailed editorial criteria is on your [guidance page](#).





BASIC REPORTING

-  Clear, unambiguous, professional English language used throughout.
-  Intro & background to show context. Literature well referenced & relevant.
-  Structure conforms to [PeerJ standards](#), discipline norm, or improved for clarity.
-  Figures are relevant, high quality, well labelled & described.
-  Raw data supplied (see [PeerJ policy](#)).

EXPERIMENTAL DESIGN

-  Original primary research within [Scope of the journal](#).
-  Research question well defined, relevant & meaningful. It is stated how the research fills an identified knowledge gap.
-  Rigorous investigation performed to a high technical & ethical standard.
-  Methods described with sufficient detail & information to replicate.

VALIDITY OF THE FINDINGS

-  Impact and novelty not assessed. Negative/inconclusive results accepted. *Meaningful* replication encouraged where rationale & benefit to literature is clearly stated.
-  Speculation is welcome, but should be identified as such.
-  Conclusions are well stated, linked to original research question & limited to supporting results.
-  Data is robust, statistically sound, & controlled.



The best reviewers use these techniques

Tip

Example

Support criticisms with evidence from the text or from other sources

Smith et al (J of Methodology, 2005, V3, pp 123) have shown that the analysis you use in Lines 241-250 is not the most appropriate for this situation. Please explain why you used this method.

Give specific suggestions on how to improve the manuscript

Your introduction needs more detail. I suggest that you improve the description at lines 57- 86 to provide more justification for your study (specifically, you should expand upon the knowledge gap being filled).

Comment on language and grammar issues

The English language should be improved to ensure that an international audience can clearly understand your text. Some examples where the language could be improved include lines 23, 77, 121, 128 - the current phrasing makes comprehension difficult.

Organize by importance of the issues, and number your points

- 1. Your most important issue*
- 2. The next most important item*
- 3. ...*
- 4. The least important points*

Please provide constructive criticism, and avoid personal opinions

I thank you for providing the raw data, however your supplemental files need more descriptive metadata identifiers to be useful to future readers. Although your results are compelling, the data analysis should be improved in the following ways: AA, BB, CC

Comment on strengths (as well as weaknesses) of the manuscript

I commend the authors for their extensive data set, compiled over many years of detailed fieldwork. In addition, the manuscript is clearly written in professional, unambiguous language. If there is a weakness, it is in the statistical analysis (as I have noted above) which should be improved upon before Acceptance.

Citation.js: a format-independent, modular bibliography tool for the browser and command line

Lars G Willighagen ^{Corresp.}

Corresponding Author: Lars G Willighagen
Email address: lars.willighagen@gmail.com

Background. Given the vast number of standards and formats for bibliographical data, any program working with bibliographies and citations has to be able to interpret such data. This paper describes the development of Citation.js (<https://citation.js.org/>), a tool to parse and format according to those standards. The program follows modern guidelines for software in general and JavaScript in specific, such as version control, source code analysis, integration testing and semantic versioning.

Results. The result is an extensible tool that has already seen adaption in a variety of sources and use cases: as part of a server-side page generator of a publishing platform, as part of a local extensible document generator, and as part of an in-browser converter of extracted references. Use cases range from transforming a list of Wikidata identifiers into a BibTeX file on the command line, to displaying RIS references on a webpage with added Altmetric badges to generating "How to cite this" sections on a blog.

Conclusions. Citation.js is a library supporting various formats of bibliographic information in a broad selection of use cases and environments. Given the support for plugins, more formats can be added with relative ease.

Citation.js: a Format-independent, Modular Bibliography Tool for the Browser and Command Line

Lars G. Willighagen¹

¹Eindhoven, The Netherlands

Corresponding author:

Lars G. Willighagen¹

Email address: lars.willighagen@gmail.com

ABSTRACT

Background. Given the vast number of standards and formats for bibliographical data, any program working with bibliographies and citations has to be able to interpret such data. This paper describes the development of Citation.js (<https://citation.js.org/>), a tool to parse and format according to those standards. The program follows modern guidelines for software in general and JavaScript in specific, such as version control, source code analysis, integration testing and semantic versioning.

Results. The result is an extensible tool that has already seen adaption in a variety of sources and use cases: as part of a server-side page generator of a publishing platform, as part of a local extensible document generator, and as part of an in-browser converter of extracted references. Use cases range from transforming a list of Wikidata identifiers into a BibTeX file on the command line, to displaying RIS references on a webpage with added Altmetric badges to generating "How to cite this" sections on a blog.

Conclusions. Citation.js is a library supporting various formats of bibliographic information in a broad selection of use cases and environments. Given the support for plugins, more formats can be added with relative ease.

INTRODUCTION

With the primary goal of scholarly publishing being the distribution of knowledge, it is important that this knowledge is actually distributed properly, which includes the accessibility and findability of these publications. One way findability has improved in the last few decades, is the transition from text-based citations to the use of Persistent IDentifiers (PIDs), with Digital Object Identifiers (DOIs) being the most common. These PIDs are linked to central stores that provide machine-readable bibliographic information.

However, there are many different stores, most with their own formats: Wikidata (Malyshev et al., 2018) and WikiCite (Taraborelli et al., 2017) have their own scheme; DataCite has DataCite eXtensible Markup Language (XML) and JavaScript Object Notation (JSON) (De Smaele et al., 2017); and CrossRef has CrossRef Unixref XML (noa, d). Similarly, most citation managers have their own formats too: Zotero and EndNote have their own schemes (Vinckevicius, 2017; noa, c), and Office Word has an XML namespace (noa, b). On top of that there are a lot of old and new formats created for a variety of reasons, like BibTeX (Patashnik, 1988), Citation Style Language (CSL) JSON (Zelle, 2012), and Research Information Systems

36 (RIS) (noa, 2012).

37 This leads us to two problems. First, citation managers need to maintain parsers for multiple formats.
38 **Second, existing managers, like Zotero, either require a server or have entirely proprietary backends.** This
39 paper introduces Citation.js, which is a set of different parsers and formatters that works anywhere. By
40 using JavaScript, it can be used as a script, a server, or in the browser. To better suit individual needs, and to
41 minimize unnecessary code, which is especially important in the browser, Citation.js is fully modularised.
42 Formats are bundled in thematic plugins, which can be installed separately. For formatted bibliographies
43 and citations, CSL templates and locales are used with citeproc-js (Zelle, 2012; Bennett et al., 2018). This
44 paper describes how Citation.js is developed, documented, tested, and released.

45 BACKGROUND

46 Software Development

47 The software was developed using modern standards: version control with Git, semantic versioning for
48 releases (Preston-Werner, 2013), open source archives on GitHub ([https://github.com/larsgw/](https://github.com/larsgw/citation.js)
49 [citation.js](https://github.com/citation-js); <https://github.com/citation-js>) and Zenodo ([https://doi.org/10.](https://doi.org/10.5281/zenodo.1005176)
50 [5281/zenodo.1005176](https://doi.org/10.5281/zenodo.1005176)), browser bundles with browserify (Halliday et al., 2018), compatible code
51 with Babel (Zhu et al., 2018), integration testing using the Travis-CI service (noa, 2018b), code linting
52 (source code analysis) with ESLint (Zakas et al., 2018) and Standard (Aboukhadijeh et al., 2018), checking
53 RegExp's for ReDOS vulnerabilities with `vuln-regex-detector` (Davis et al., 2018), and detailed
54 documentation with JSDoc (Williams et al., 2018).

55 The development process took place with Node.js and npm. First off, any changes would be linted and
56 tested with the aforementioned tools. Bugs or new features can also warrant the introduction of new test cases.
57 If the changes work properly, they are then committed into the version control. If the changes warrant a new
58 release, or if enough changes have piled up for a new release, the change log is updated. Updating the version
59 in the package metadata automatically triggers the linters and test runners, preventing accidental mistakes.
60 Afterwards, publishing the package to npm automatically triggers the generation of files necessary for the
61 package. The scripts used for this are described in [https://github.com/larsgw/citation.js/](https://github.com/larsgw/citation.js/blob/90cd68c/CONTRIBUTING.md#installing)
62 [blob/90cd68c/CONTRIBUTING.md#installing](https://github.com/larsgw/citation.js/blob/90cd68c/CONTRIBUTING.md#installing).

63 Libraries and others

64 Apart from tools used for development, Citation.js also uses a number of runtime libraries. Their function
65 and the reason for using them is explained below.

66 **@babel/polyfill** is a runtime library which fills in support for modern APIs on older platforms. It
67 comes with the use of Babel to transform modern syntax for older platforms (Zhu et al., 2018).

68 **citeproc** is the official CSL formatting library in JavaScript (Bennett et al., 2018; noa, a).

69 **commander** is a utility library, only used for the Command Line Interface (CLI). It parses the command
70 line arguments and generates documentation (Holowaychuk et al., 2018).

71 **isomorphic-fetch** is a specific polyfill, a library filling in support, for the Fetch Application Pro-
72 gramming Interface (API), a modern way of requesting web resources. It works in both Node.js and
73 browsers (Andrews et al., 2018).

74 **sync-request** is a way to request web resources synchronously. While performing such operations
75 synchronously is advised against in JavaScript, it is still useful for non-production scientific scripts,
76 and demos (Lindesay et al., 2018).

77 **wikidata-sdk** is a utility library for working with the Wikidata API (Lathuilière et al., 2018; Vrandečić
78 and Krötzsch, 2014)

79 .

80 RESULTS

81 Implementation

82 Citation.js employs a number of ways to achieve a balance between function and ease of use. The program
83 consists of three major parts: the bibliography interface, code handling input parsing, and code handling
84 output formatting. The bibliography interface itself is quite simple; it mainly acts as a wrapper around the
85 parsing and formatting parts. These two parts behave in a modularised way, with a common plugin system.

86 *Input parsing*

87 Input parsing works by registered input formats. These registrations include an optional type recognizer
88 and a synchronous and/or an asynchronous function transforming the input into a format closer to the final
89 format: CSL-JSON. The new input can then be tested again, and will be parsed iteratively until the final
90 format is reached. Plugin authors are encouraged to create input parsers with as small steps as possible, to
91 allow users to input a variety of different formats.

92 Type recognition is done with a search tree. First of all, types are ordered by the data type of the
93 input. This is one of: `String` (unparsed text, identifiers, etc.), `SimpleObject` (standard JavaScript
94 `Object`), `Array` (a possibly non-uniform list of inputs), `ComplexObject` (other non-literal values) and
95 `Primitive` (numbers, null, undefined). The data type can be inferred from other format specifications
96 in some cases. Types can also be specified to be a more specific version of something else. For example, a
97 DOI Uniform Resource Locator (URL) is also a normal URL, but should be parsed differently, **namely with**
98 **content negotiation.**

99 Types can then provide a list of predicates, testing if input belongs to that format. To avoid code repetition
100 and make plugin registration code easier to read, certain common tasks can also be accomplished using
101 shortcuts. These shortcuts include testing text against a `RegExp` pattern, checking for certain properties and
102 checking for the value of elements in an array. These properties can also eliminate the need for an explicit
103 data type: for example, if a `RegExp` is provided, input can be expected to be a `String`.

104 *Output formatting*

105 Output formatting is less complicated. Users and developers only have to provide the identifier of the
106 formatter. Further customization can then be done by providing options, which are automatically forwarded
107 to the formatter. This allows the CSL plugin to take in options specifying the template and locale, for
108 example.

109 *Plugin system*

110 Apart from being able to add input and output formats and schemes on their own, it is also possible to add
111 them in a thematically linked plugin. For example, a BibTeX plugin might consist of a parser for `.bib`
112 files, a parser for the resulting BibTeX-schemed JSON, and a output formatter to create BibTeX from other
113 sources as well. This plugin could then be combined with, for example, a Bib.TXT plugin, resulting in

```
1 let Cite = require('citation-js')
2
3 Cite.plugins.add('bibtex', {
4   input: {
5     '@bibtex/text': {
6       parseType: { ... },
7       parse (text) { ... }
8     },
9     '@bibtex/object': {
10      parseType: { ... },
11      parse (text) { ... }
12    }
13  },
14
15  output: {
16    bibtex (data, options) {
17      ...
18    }
19  },
20
21  config: {
22    labelForm: ['author', 'title', 'issued']
23  }
24 })
25
26 let bibtexConfig = Cite.plugins.config.get('bibtex')
27 bibtexConfig.labelForm = ['author', 'issued', 'year-suffix']
```

Figure 1. Possible structure of a plugin for BibTeX. In this example package, line 1 loads Citation.js and lines 2-24 adds the plugin. This plugin consists of two input formats (4-13), one output format (15-19) and configuration options (21-23). Lines 26-27 show how this configuration would be used. Some code is omitted for the sake of clarity, and is replaced with ellipsis (. . .).

114 a JavaScript package or module, which could be published in package managers like npm. Code for this
115 plugin would look like Fig. 1.

116 For configuring plugins there is also a `config` option. As an example a `labelForm` option is added,
117 which could control the way the BibTeX output formatter generates labels. Users of this plugin can then
118 retrieve and modify this configuration. It is also possible to offer internal functions this way, for more
119 fine-grained control.

120 ***Bibliography interface***

121 The methods for parsing input and formatting output are also included in a general class, `Cite`. Class
122 instances also have access to opt-in version control – changes are tracked if an explicit flag is passed – and
123 sorting. The latter currently does not have effect on CSL bibliographies unless set with the `nosort` option,
124 as the styles define their own sorting method.

125 ***Supported formats***

126 Table 1 shows the formats supported by Citation.js at the moment.

127 **Distribution**

128 ***Browser use***

129 For in-browser use, there is also a standalone JavaScript file available. This includes dependencies. This
130 bundle is built automatically when publishing, and is available through a number of Content Delivery

Table 1. Input and output format support. This table only shows general support. For example, the "Wikidata" format is both used for Wikidata identifiers and Wikidata API results.

Format	Input	Output
BibJSON	x	
BibTeX	x	x
Bib.TXT	x	x
CSL	(JSON)	x
DOI	x	
RIS		x
Wikidata	x	

131 Networks (CDNs) that automatically distribute npm packages. The `Cite` class can then be imported and
 132 used just as the npm package, barring browser limitations.

133 For simple use cases like inserting static bibliographies, a separate tool, `citation.js-replacer`,
 134 was developed. When included on a page, this replaces every HyperText Markup Language (HTML) element
 135 matching a certain selector, with a bibliography.

136 Figure 2 shows an example of another use case. For example, the basic use can be extended to add
 137 additional information to references, such as an Altmetric (Adie and Roe, 2013) score icon or Dimensions
 138 citation count (Thelwall, 2018).

139 *npm package*

140 `Citation.js` is published as an npm package on the main npm registry, as `citation-js`. Use of the package
 141 is the same anywhere, apart from platform limitations. For example, synchronous requests for web resources,
 142 used to get metadata for DOIs, is limited on Chrome as discussed in Willighagen (2017b). Also, the Node.js
 143 platform, not being a browser, doesn't have access to the Document Object Model (DOM), and so can't
 144 easily use HTML elements as input or output.

145 Separate components, including formats not included in the standard configuration are available under
 146 the `@citation-js` scope.

147 Use cases for the npm package include using it when generating content (either at runtime or for static
 148 websites) like PubPub (Shihpar and Rich, 2018), and setting up APIs (Willighagen, 2017a). It is also useful
 149 for converting metadata when text mining. For example, BibJSON is one of the input formats, and can then
 150 be converted to BibTeX or formatted. All citations to GitHub projects were created with a simple script
 151 running `Citation.js`.

152 *CLI use*

153 Simple one-time conversions, with no extensive customization, can also be done with Command Line
 154 Interface (CLI). The command requires Node.js and npm and can be installed by running `npm install`
 155 `-global citation-js`, which may require root privileges depending your setup. Alternatively, any
 156 commands can be prefixed with `npmx` instead. The command can read files with `-i`, `-input`, input text
 157 with `-t`, `-text` or via standard in. Output can be configured with a number of options detailed in the man
 158 file, also available by running with the `-h`, `-help` option. Any output is then written to a file with `-o`,
 159 `-output` and else redirected to standard out. Additional logging and errors can be found on standard error.

```
1 <html>
2 <head>
3   <!-- Altmetric widget code --> <script
4     ↪ src="https://dlbvxh8uaslmw7.cloudfront.net/assets/embed.js"></script>
5   <script src="https://cdn.jsdelivr.net/npm/citation-js"></script>
6 </head>
7 <body>
8   <div id="element"></div>
9   <script>
10     window.onload = async function () {
11       let Cite = require('citation-js')
12       let cite = await Cite.async('10.1371/journal.pone.0185809')
13
14       let bibliography = cite.format('bibliography', {
15         format: 'html',
16         append ({DOI}) {
17           return ` <span class="altmetric-embed" data-badge-type="badge" data-doi="${DOI}"></span>`
18         }
19       })
20
21       let element = document.getElementById('element')
22       element.innerHTML = bibliography
23       _altmetric_embed_init()
24     }
25   </script>
26 </body>
27 </html>
```

Figure 2. Basic use, including appending data to formatted bibliography entries. Here, line 3 loads the Altmetric widget code, line 4 loads the library, line 10 imports `Cite`, and line 11 creates an interface for a bibliography with one entry, with metadata from a DOI. Lines 13-18 render the bibliography, with line 14 setting the output to HTML and lines 15-17 appending an Altmetric widget to the entry. Lines 20-21 show the output on the page. Lines 9 and 22 are to avoid race conditions in DOM access. Line 3 and 22 initialize the Altmetric badge. In the example, (Hallmann et al., 2017) is used.

160 **Integrations**

161 The Citation.js npm package can also be used as a library to create integrations with, among other things,
162 word processing systems. For example, ReLaXed (Zulko et al., 2018) integrates Citation.js into the Pug
163 templating language to generate citations when creating Portable Document Format (PDF) documents,
164 and the npm package `citation-js-showdown` was created as a demo on how to introduce syntax for
165 citations in markdown.

166 **Performance**

167 The performance of the Citation.js package has been analyzed on a number of different platforms. Between
168 browsers, compiling the script and importing the library takes about 120 ms, compiling itself taking a little
169 less than half of that. Node.js on the other hand takes about 1 s to initialize, both when the source consisting
170 of multiple files is imported, and when a bundle is imported. This is possibly because Chrome caches
171 compiled JavaScript reducing the compiling times from around 50 ms to about 8 ms, as is explained in Alle
172 (2018).

173 The results were obtained with the default settings for each of the platforms. In Chrome, this was
174 achieved by using guest mode, while Node and Firefox were not configured to begin with.

175 As shown in Fig. 4 and Table 2, time taken to import the library mainly consists of importing
176 `@babel/polyfill`. This is because adding the polyfills requires repeated feature detection. After

Hallmann, C. A., Sorg, M., Jongejans, E., Siepel, H., Hofland, N., Schwan, H., ... de Kroon, H. (2017). More than 75 percent decline over 27 years in total flying insect biomass in protected areas. *PLOS ONE*, 12(10), e0185809. <https://doi.org/10.1371/journal.pone.0185809>



Figure 3. Result of the code in Fig. 2. Bibliography consisting of Hallmann et al. (2017) in APA style. Note the Altmetric badge at the end.

177 that the actual code is imported in two parts. In the first part, where core functionality like the Cite interface
178 is loaded, the main culprit is `addTypeParser`, with 0.13 ms per call on average. In the second part,
179 loading output-related code, importing `citeproc-js` takes the longest with a single call of 2.82 ms. Note
180 that that Firefox uses Just-In-Time (JIT) compilation, compiling pieces of code when they are used a lot.

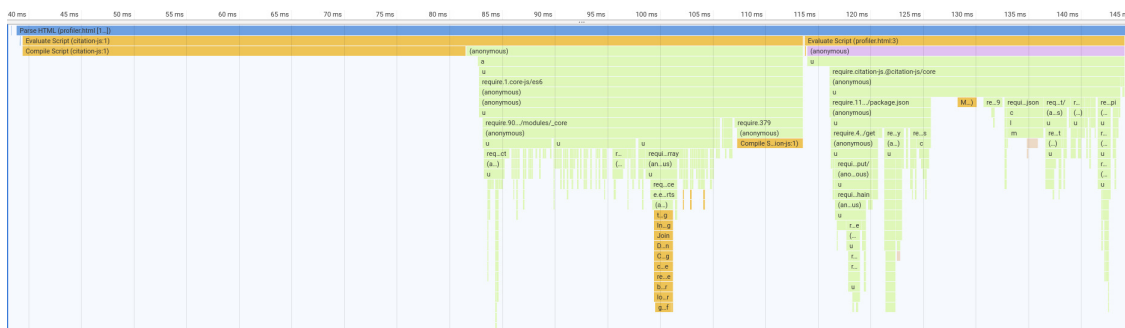
181 While code execution is one part, one should also look into the file size. This is especially important in
182 the browser, which has to fetch the library when loading the page.

183 DISCUSSION

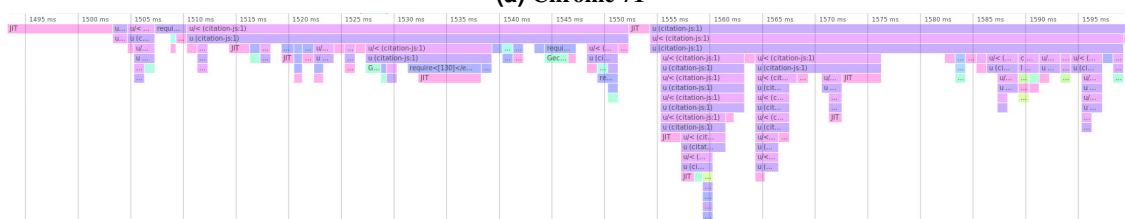
184 Use of formal grammars for parsing

185 Apart from more common formats like JSON, XML and YAML (YAML Ain't Markup Language), Citation.js
186 has to parse a number of text formats with syntax specific to that format, like BibTeX and RIS. While one can
187 use standard or even built-in parsers for common formats, that is usually not possible for the latter formats.
188 To solve this, one can employ formal grammars, which can be translated into code parsing and validating
189 input. Examples of libraries working with grammars are PEG.js (Majda et al., 2018) and nearley.js (Kartik
190 et al., 2018). Creating grammars has the benefits of not having to write and maintain code validating and
191 parsing input, and having a readable grammar instead of a complex program file.

192 However, there are also drawbacks. Generating these grammars requires an extra build step, which in
193 the case of Citation.js cannot be integrated with the preceding step due to a lack of appropriate tooling. On
194 top of that, early tests have shown generated code to have poor performance or large size, and in the case
195 of nearley.js requires a runtime library. Therefore, it may be preferable to write custom parsers in some
196 cases. For example for RIS, which has no balanced brackets or quotes, and does not need much more than
197 simply iterating over the individual lines.



(a) Chrome 71



(b) Firefox 64

Figure 4. Initialization performance results on different platforms. Actual timings may vary depending on the device, operating system and cache. Note that Chrome starts with 40 ms of compiling time that is cached on subsequent runs. Firefox compiles JIT, while the code is running. Both graphs show three parts, one loading polyfills from @babel/polyfill taking up half the loading time, followed by two parts mainly loading core functionality and plugins respectively. Profiling data is available in supplemental information.

198 Converting between formats and standardized crosswalks with linked data

199 Converting input data like parsed BibTeX, BibJSON or Wikidata API results into another format and back
 200 can get very repetitive in terms of code. Yet, there are still cases where special handling is needed. Since
 201 different formats call for different needs, each plugin has developed its own system to deal with this. Unifying
 202 this into a single, performant, reusable and developer-friendly system would be preferable.

203 To convert one data format (or scheme) to another, a number of different things must be done. First of
 204 all, for most properties a simple mapping suffices: `title` in BibTeX refers to the same concept as it does
 205 in BibJSON and CSL-JSON. On top of that, the property coincides with `TI` in RIS. This mapping could
 206 come in the form of a JSON Linked Data (JSON-LD) context, as done by CodeMeta (Jones et al., 2017).

207 Second, the data format needs to be converted. While `title` in BibTeX can have formatting in the form
 208 of `TeX`, `title` in CSL-JSON uses a subset of HTML for formatting, and `TI` in RIS does not have formatting
 209 at all. Properties can also have different data types. In CSL-JSON, `author` is a list of objects, while
 210 `authors` in BibTeX, which describes the same concept, is serialized text delimited by " and ". Both of
 211 these examples could be handled by defining two converters for each pair of formats, one for converting
 212 from A to B and one for converting back. It would also be possible to convert every format to a central
 213 format C, and convert C to every format. This is what Citation.js currently does with CSL-JSON as the

Table 2. Browser bundle breakdown. Running time is the time it takes to import that part of the script with browserify `require` in the Chrome data set. Note that a big part of the `plugin-csl` "own" code is serialized styles and locales from the CSL repository. Minified the code is 702 kB, which is reduced to 177 kB with gzip and 164 kB with Brotli, both with default compression levels.

Part		Size			Running time
		Own	Dependencies	Total	
Citation.js	Backport	5.9 kB	-	5.9 kB	5.9 ms
	core	99.7 kB	33.9 kB	133.5 kB	8.3 ms
	plugin-bibjson	7.6 kB	-	7.6 kB	3.5 ms
	plugin-bibtex	42.9 kB	-	42.9 kB	2.6 ms
	plugin-csl	87.0 kB	461.8 kB	548.9 kB	3.2 ms
	plugin-doi	6.6 kB	-	6.6 kB	0.6 ms
	plugin-ris	11.1 kB	-	11.1 kB	0.5 ms
	plugin-wikidata	22.1 kB	40.4 kB	62.4 kB	2.2 ms
	name	16.6 kB	-	16.6 kB	1.1 ms
	date	7.3 kB	-	7.3 kB	0.2 ms
Additional	@babel/polyfill	-	197.3 kB	197.3 kB	32.0 ms
	browserify	-	6.8 kB	6.8 kB	0.2 ms
Total		306.8 kB	740.1 kB	1046.8 kB	60.3 ms

214 central format. It is however important that format C can hold as much information as could be represented
 215 in any other format, as to prevent information loss when converting between two formats. This is currently
 216 the case with the software entry type, which does not exist in CSL 1.0.1, and is represented by the `book`
 217 type by convention.

218 Third, there might not be a one-to-one mapping between properties. For instance, `pages` in CSL-JSON
 219 maps to both `start` and `end` in Citation File Format (CFF) (Druskat et al., 2018). Similarly, `pages`,
 220 `issue`, `volume` and `ISSN` are all top-level properties in CSL-JSON, while the corresponding properties
 221 in Wikidata are proposed to be nested in the journal property. If the nested values are deserialized, this
 222 could be expressed in JSON-LD contexts. However, in the first example it cannot, since JSON-LD cannot
 223 distinguish between parts of strings.

224 Lastly, some mappings are context-dependent. For example, consider the CSL-JSON properties `author`
 225 and `reviewed-author` in relation to the RIS properties `author` (AU) and `reviewer` (usually C4). In
 226 normal entries, AU maps to `author`. However, if the entry being converted is a review, AU maps to
 227 `reviewed-author` while `author` maps to C4.

228 In conclusion, a number of different things need to be taken into account when writing a system for
 229 crosswalks. While a JSON-LD context would scale very well without a central format, most cases restrict
 230 the usefulness. Alternatively, a custom system could be developed that defines as much mapping as possible
 231 to and from a central format, with special cases for the third and last restrictions. Additional mappings to
 232 common formats, to cover properties missing from the central format could be added, with caution not to
 233 create critical dependencies on other formats. Or, given the complexity of the systems, it may be best to
 234 create for each format, allowing optimization for the special cases of that format.

235 Use of GraphQL for API queries

236 With REpresentational State Transfer (REST) APIs comes the problem of over-fetching and under-fetching.
237 This means that when fetching a resource it may contain too much unneeded information, require additional
238 calls to the API, or both. This causes unnecessary load on both the client and the server, as both have to
239 process more calls with more network bandwidth.

240 Using GraphQL APIs circumvents this problem by allowing the client to specify exactly what data it
241 needs (noa, 2018a). However, not many endpoints have a GraphQL API available, so this solution cannot be
242 used everywhere.

243 Support for additional formats

244 Apart from the formats currently supported in Citation.js (see Table 1), there are plans to include more
245 formats such as EndNote, Machine-Readable Cataloging (MARC) (Avram, 2003), the Zotero scheme and
246 Office XML. These will be published in thematic plugins. For example, formats used to describe software
247 projects are joined in the plugin @citation-js/plugin-software-formats. These formats will
248 also include linked data scraped from web pages.

249 Scraping from source versus fetching from central stores

250 When getting data from, for example, the Wikidata API or scraped from a web page, that data may be
251 incomplete. However, if part of the data you get is the DOI linked to the entity queried, you could amend
252 that data with data fetched from a central store like Crossref or DataCite. Due to difficulties with prioritizing
253 data sources and non-trivial merging conflicts this has not been implemented yet. Additionally, if the user
254 specifically requests data from a specific API, it can be assumed they want that specific data to be used.

255 CONCLUSIONS

256 With the here introduced Citation.js we now have a library that supports bibliographic information in various
257 formats, from multiple sources. By using JavaScript we ensure it can be used in a wide variety of use
258 cases in the web browser, on the command line, and in a server environment. The tool is developed using
259 modern approaches and released via the npm network and archived on GitHub and Zenodo. In addition to
260 machine-readable formats such as BibTeX and RIS, the support for CSL templates ensures that citations and
261 bibliographies can be formatted in many textual representations. Additional content can be easily added
262 to those representations, such as Altmetric icons. The support for plugins allows additional formats to be
263 integrated with relative ease, and without the need of a central repository managing those plugins.

264 ACKNOWLEDGMENTS

265 Thanks to JS.org for providing the (sub)domain name for the homepage of Citation.js. Thanks to the
266 many people submitting bug reports, pull requests, and other kinds of feedback during the development of
267 Citation.js.

268 REFERENCES

- 269 CSL Processors - Developers. [https://citationstyles.org/developers/#/
270 csl-processors.](https://citationstyles.org/developers/#/csl-processors)
- 271 DocumentFormat.OpenXml.Bibliography Namespace. [https://docs.microsoft.com/en-us/
272 dotnet/api/documentformat.openxml.bibliography.](https://docs.microsoft.com/en-us/dotnet/api/documentformat.openxml.bibliography)

- 273 EndNote X6 Help User Guide (For Macintosh). <https://web.archive.org/web/20180127234028/http://endnote.com/sites/en/files/support/endnotex6machelp.pdf>.
- 274
275
- 276 UNIXREF query output format. <http://support.crossref.org/hc/en-us/articles/214936283-UNIXREF-query-output-format>.
- 277
- 278 (2012). RIS Format Specifications. https://web.archive.org/web/20120526103719/http://refman.com/support/risformat_intro.asp.
- 279
- 280 (2018a). GraphQL. <https://facebook.github.io/graphql/June2018/>.
- 281 (2018b). Travis CI - Test and Deploy Your Code with Confidence. <https://travis-ci.org/>.
- 282 Aboukhadijeh, F., Flettre, D., developerjin, Yang, A., Unneback, L., Lavieri, D., Sta. Cruz, R., Cousens, D., Greenkeeper, Comnes, B., Littauer, R., Nassri, A., Liu, W., Colvin, J., Barros, R., Robin, Emanuele, Spac, J., Watson, T., Mathur, T., Varela, P., Reggi, T., Wuyts, Y., Chen, S., M., A., Horst, B., Charlie, and Prigara, E. (2018). standard/standard. <https://github.com/standard/standard>.
- 283
284
285
- 286 Adie, E. and Roe, W. (2013). Altmetric: enriching scholarly content with article-level discussion and metrics. *Learned Publishing*, 26(1):11–17.
- 287
- 288 Alle, M. (2018). Improved code caching. <https://v8.dev/blog/improved-code-caching>.
- 289 Andrews, M., Tschinder, D., Knuth, E., Josiah, Cirkel, K., Melnikow, P., Evans, R., Giles, S., and Simeon (2018). matthew-andrews/isomorphic-fetch. <https://github.com/matthew-andrews/isomorphic-fetch>.
- 290
291
- 292 Avram, H. D. (2003). Machine-readable cataloging (MARC) program. In *Encyclopedia of Library and Information Science*, volume 3, pages 1712–1730. CRC Press, second edition.
- 293
- 294 Bennett, F., Adam, Sifford, D., Maloney, C., Ruf, D., Mower, M., C-P, F., SM, M., and Zelle, R. M. (2018). Juris-m/citeproc-js. <https://github.com/Juris-M/citeproc-js>.
- 295
- 296 Davis, J. C., Coghlan, C. A., Servant, F., and Lee, D. (2018). The impact of regular expression denial of service (ReDoS) in practice: an empirical study at the ecosystem scale. In *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 246–256, Lake Buena Vista, FL, USA. ACM Press.
- 297
298
299
- 300 De Smaele, M., Starr, J., Ashton, J., Birt, N., Dietiker, S., Elliott, J., Fenner, M. Hatfield Hart, A., Hugo, W., Jakobsson, S., Bernal Martínez, I., Rücknagel, J. Yahia, M., Ziedorn, F., and Zolly, L. (2017). DataCite Metadata Schema Documentation for the Publication and Citation of Research Data v4.1. <https://schema.datacite.org/meta/kernel-4.1/index.html>.
- 301
302
303
- 304 Druskat, S., Chue Hong, N., Haines, R., and Baker, J. (2018). Citation File Format (CFF) - Specifications. <https://zenodo.org/record/1003149>.
- 305
- 306 Halliday, J., Suarez, A., Kooi, R., Aboukhadijeh, F., Denicola, D., McCarthy, J., Lindesay, F., Popp, A., Stock, T., Lorenz, T., Duff, J., Shtylman, R., Kennedy, H., Harband, J., Patrick, Mao, S., Anatoliy, Metcalf, C., McConnell, C., Kastner, D., Govett, D., Wright, J., Ewald, J., Parodi, A., Comnes, B., Macabiau, C., Gozalishvili, I., and Hanson, J. (2018). browserify/browserify. <https://github.com/browserify/browserify>.
- 307
308
309
310
- 311 Hallmann, C. A., Sorg, M., Jongejans, E., Siepel, H., Hofland, N., Schwan, H., Stenmans, W., Müller, A., Sumser, H., Hörren, T., Goulson, D., and de Kroon, H. (2017). More than 75 percent decline over 27 years in total flying insect biomass in protected areas. *PLOS ONE*, 12(10):e0185809.
- 312
313
- 314 Holowaychuk, T., Li, Z., Abe, T., Koutnik, R., Geraghty, T., Lukasavage, T., Allen, D., Agius, A., Petersen, J., Neeman, I., De Bollivier, S., Nichols, A., Lyons, P., Sorohan, B., Geisendörfer, F., Yaroshevich, A., Franzoia, A., Ruf, D., George, J., Robertson, J., Hamlet, J., Brooks, M., Thomas, R., Vanesyan, R., Ilia, Buathier, Q., and Nitta, A. (2018). tj/commander.js. <https://github.com/tj/commander.js>.
- 315
316
317

- 318 Jones, M. B., Boettiger, C., Mayes, A. C., Slaughter, P., Niemeyer, K., Gil, Y., Fenner, M., Nowak, K., Hahnel,
319 M., Coy, L., Allen, A., Crosas, M., Sands, A., Hong, N. C., Cruse, P., Katz, D., and Goble, C. (2017).
320 CodeMeta: an exchange schema for software metadata. <https://raw.githubusercontent.com/codemeta/codemeta/2.0/codemeta.jsonld>.
321 <https://raw.githubusercontent.com/codemeta/codemeta/2.0/codemeta.jsonld>.
- 322 Kartik, Radvan, T., Stewart, A., Corbin, J. T., Windels, R., Marinov, B., Emanuel, K., Viet, L. Q., Itzhaky,
323 S., alex, Litvin, N., Olmsted, S., Hunter, C., Edelman, J., Kanefsky, B., Jake, Aukia, J., Kemp, K. J.,
324 Ljunglöf, P., Rose, R., Hildebrandt, S., Trefz, A., Rosenzweig, A., Gunderson, B., Meadors, C., Bertolini,
325 F., Victorio, F., and Quigley, J. (2018). kach/nearley. <https://github.com/kach/nearley>.
- 326 Lathuilière, M., Voß, J., Simantov, M., Willighagen, L., Roberts, L., and offirmo (2018). maxlath/wikidata-sdk.
327 <https://github.com/maxlath/wikidata-sdk>.
- 328 Lindesay, F., Joppi, D. H., Kannan, S., Irving-Beer, A., Double, C., Dascalescu, D., Zaharee, D., Hoffmann,
329 D., Krems, J. O., Hong, J., Bílek, K., Willighagen, L., and Zoltu, M. (2018). Forbeslindesay/sync-request.
330 <https://github.com/ForbesLindesay/sync-request>.
- 331 Majda, D., Ryuu, F.-z., Breault, A., Ruciński, R., felix, <https://github.com/josephfrazier>, Sampson, A.,
332 Tavakoli, A., Vrána, J., Blank, J., Pirsch, F., Ramjiawan, A., Mimms, A., Almad, Neculau, A., Kutil,
333 B., Hearon, C., Davies, J., Doersing, N., Brandt, P., Lukasavage, T., and chungpu (2018). pegjs/pegjs.
334 <https://github.com/pegjs/pegjs>.
- 335 Malyshev, S., Krötzsch, M., González, L., Gonsior, J., and Bielefeldt, A. (2018). Getting the Most Out of
336 Wikidata: Semantic Technology Usage in Wikipedia’s Knowledge Graph. In Vrandečić, D., Bontcheva,
337 K., Suárez-Figueroa, M. C., Presutti, V., Celino, I., Sabou, M., Kaffee, L.-A., and Simperl, E., editors, *The*
338 *Semantic Web – ISWC 2018*, volume 11137, pages 376–394. Springer International Publishing, Cham.
- 339 Patashnik, O. (1988). BibTeXing. [http://mirrors.ctan.org/biblio/bibtex/base/](http://mirrors.ctan.org/biblio/bibtex/base/btxdoc.pdf)
340 [btxdoc.pdf](http://mirrors.ctan.org/biblio/bibtex/base/btxdoc.pdf).
- 341 Preston-Werner, T. (2013). Semantic Versioning 2.0.0. <https://semver.org/>.
- 342 Shihipar, T. and Rich, T. (2018). PubPub: Open publishing. <https://pubpub.org>.
- 343 Taraborelli, D., Pintscher, L., Mietchen, D., and Rodlund, S. (2017). WikiCite 2017 report. https://figshare.com/articles/WikiCite_2017_report/5648233/3.
- 344 Thelwall, M. (2018). Dimensions: A competitor to Scopus and the Web of Science? *Journal of Informetrics*,
345 12(2):430–435.
- 346 Vinkevicius, A. (2017). Zotero to CSL extension and mappings. [https://aurimasv.github.io/](https://aurimasv.github.io/z2csl/typeMap.xml)
347 [z2csl/typeMap.xml](https://aurimasv.github.io/z2csl/typeMap.xml).
- 348 Vrandečić, D. and Krötzsch, M. (2014). Wikidata: a free collaborative knowledgebase. *Communications of*
349 *the ACM*, 57(10):78–85.
- 350 Williams, J., Mathews, M., Frank, J., Wrzeszcz, R., Blank, B., Colter, M., Schaub, T., Hayes, L., Haagsman,
351 E., Pozin, K., Kim, M., Phasmal, Harrtell, B., Kerns, B., Chambers, D., Schonning, N., Droogmans, P.,
352 Taylor, R., Tutt, B., Parks, C., Locke, K., Dubeau, L.-D., Voyer, V., Nicksay, A., Kienzle, S., Nison, M.,
353 and Wytębowicz, T. (2018). jsdoc3/jsdoc. <https://github.com/jsdoc3/jsdoc>.
- 354 Willighagen, L. (2017a). Citation.js: Endpoint on RunKit. [https://larsgw.blogspot.com/2017/](https://larsgw.blogspot.com/2017/08/citationjs-endpoint-on-runkit.html)
355 [08/citationjs-endpoint-on-runkit.html](https://larsgw.blogspot.com/2017/08/citationjs-endpoint-on-runkit.html).
- 356 Willighagen, L. (2017b). Make a synchronous redirectable CORS request
357 in Chrome. [https://stackoverflow.com/questions/45238537/](https://stackoverflow.com/questions/45238537/make-a-synchronous-redirectable-cors-request-in-chrome)
358 [make-a-synchronous-redirectable-cors-request-in-chrome](https://stackoverflow.com/questions/45238537/make-a-synchronous-redirectable-cors-request-in-chrome).
- 359 Zakas, N. C., Volodin, I., Katz, T., Singh, G., Nagashima, T., alberto, Mills, B., Partington, K., Ficarra, M.,
360 Cataldo, K., Kaya, B. Y., Myers, I. C., Schreck, M., VanSchooten, I., DuVall, M., Pedrotti, M., Balocco,
361 V., Aliaksei, Rajavuori, J., Anson, D., Cochard, G., Allardice, J., Fang, P., Hom, V., Pool, J., Harband, J.,
362

- 363 @scriptdaemon, and Vidal, J. R. (2018). eslint/eslint. <https://github.com/eslint/eslint>.
- 364 Zelle, R. M. (2012). CSL 1.0.1 Specification. [http://docs.citationstyles.org/en/1.0.1/](http://docs.citationstyles.org/en/1.0.1/specification.html)
365 [specification.html](http://docs.citationstyles.org/en/1.0.1/specification.html).
- 366 Zhu, H., Smyth, L., Ng, B., Haverbeke, M., Masad, A., Tschinder, D., Stepanyan, I., Jamie, Ridgewell,
367 J., Ribaud, N., Sauleau, S., Yavorsky, A., Hanson, A., McCarthy, J., Burzyński, M., Donovan, B.,
368 Goldman, S., DiGioia, J., Zilberman, M., Franco, J., Bynens, M., Kappert, L., Kushwaha, P., Bedford,
369 G., Newman, B., Gonçalves, A. A. S., Cataldo, K., and Abramov, D. (2018). babel/babel. [https:](https://github.com/babel/babel)
370 [//github.com/babel/babel](https://github.com/babel/babel).
- 371 Zulko, Drew, Townsend, T., Ruf, D., Li, J., Mitra, N., Forrest, G., Koska, K., David, and Idrovira (2018).
372 Relaxedjs/relaxed. <https://github.com/RelaxedJS/ReLaXed>.