Frank Bennett <biercenator@gmail.com>

# CSL processor news

**Frank Bennett <biercenator@gmail.com>** **Wed, Mar 4, 2009 at 10:41 AM**
To: zotero-dev <zotero-dev@googlegroups.com>

I've been doing some work on a new CSL processor (citeproc-js), and have made some progress. The next academic term is approaching, though, and I'll be battening down the work on citeproc-js over the next few days. I'll be pretty much leaving the code alone until sometime during the summer, but I don't claim ownership of it, and any work by others on the project will be very welcome as far as I'm concerned. (In fact, it's probably better for the long term if I'm not the primary maintainer. I'm a hobbyist, my skill level is not that high, and my ability to focus on programming issues varies with the season.) Before downing tools, I'll go through the code to update the comments and bring them into line with the state of the code. The test suites don't show much organization, but I'll probably leave those alone for the present.

It's been an exciting ride over the past month. There's a lot still to do, but most of the seriously worrisome issues have been cleared. Here are some of the highlights:

- The commented code is available online at: http://gsl-nagoya-u.net/http/pub/citeproc-js-doc/index.html

- The sources are available at: http://xbiblio.svn.sourceforge.net/viewvc/xbiblio/citeproc-js/

- There are 93 tests in the test suite covering the work to date, all of which pass.

- CSL.Build and CSL.Configure assemble a style object containing only the functions needed to render the requested style (no more if/then/else over the entire CSL language to determine the operation to perform).

- CSL.Render produces output from the compiled style. End-to-end input/output testing of added functionality is now possible.

- Locale loading and overloading from the CSL style both work correctly.

- Conditional branching is implemented efficiently and works correctly.

- Macros are resolved in CSL.Build and vanish, simplifying the rest of the code base.

- The E4X XML parser has been encased in a wrapper, and the wrapper has been tested against both E4X and a homebrew Javascript-only parser bundled with the code. Support for other parsers can be added with very little additional effort.

- The beginnings of a similarly modular system for retrieving Item data is in place, which will permit the engine to accept item keys as well as full Item objects as input.

- Persistent state awareness across a series of cites (i.e. within a "citation") is available, and can be exploited to control capitalization and splicing.

Here are some things that still need to be done:

- Standard end-to-end test fixtures for CSL styling are needed, and machinery to digest and apply them needs to be built.

- A disambiguation/sorting registry needs to be completed and tested, and integrated into CSL.Render, and again tested. Some framework code for this is in tests/test_speed.js (the CSL.Registry code is wrong, and should be redone from scratch, using the test_speed.js code as a rough model).

- Separate opt/token areas, similar to state.citation and state.bibliography, need to be established for producing sort keys and for the disambiguation of entries according to parameters that still need to be defined. These areas will be populated with tokens by CSL.Build. For sort keys, this will be done with a build method to the sort tag, and for disambiguation through a function invoked at the close of /citation, after the various disambiguate attributes have been collected from the style (the description is a little opaque, but it should make sense when you look through the code). These two special token renderers will be needed to get the registry going.

- Name formatting needs to be implemented. I'm guessing that 90% or more of the remaining work to implement the engine lies here and in disambiguation.

- Date formatting needs to be implemented, together with the localized dates proposal recently added (i think) to CSL.

- Various options and special formatting attributes need to be implemented.

- All of the additions above need to be rigorously tested, so that the engine doesn't blow up in our faces when we turn on the switch (!).

I'll be following the list while I'm "away", so feel free to post questions if you look at the code and something doesn't make sense.

Frank Bennett
Nagoya
2009-03-04

---