1 DONE Exporting numbered citations in html with unsorted numbered bibliography

In this post we illustrated a simple export of org-ref citations to html. This was a simple export that simply replaced each citation with a hyperlink. Today we look at formatting them with numbers, and having a numbered bibliography. This will take multiple passes to achieve. One pass to get the citations and calculate replacements, and one pass to replace them.

This text is just some text with somewhat random citations in it for seeing it work. You might like my two data sharing articles.[1,2] We illustrate the use of org-mode in publishing computational work,[3–5] experimental[6] and mixed computational and experimental work.[7,8] This example will correctly number multiple references to a citation, e.g.[1] and.[2]

This post is somewhat long, and the way I worked it out is at the end (Long appendix illustrating how we got the code to work). The short version is that we do some preprocessing to get the citations in the document, calculate replacement values for them and the bibliography, replace them in the org-buffer *before* the export in the backend (html) format we want, and then conclude with the export. This is proof of concept work.

The main issues you can see are:

1. Our formatting code is very rudimentary, and relies on reftex. It is not as good as bibtex, or presumably some citation processor. Major improvements would require abandoning the reftex approach to use something that builds up the bibliography entry, allows modification of author names, and accomodates missing information gracefully.

2. The bibliography contents reflect the contents of my bibtex file, which is LaTeX compatible. We could clean it up more, by either post-processing to remove some things like escaped &, or by breaking compatibility with LaTeX.

3. The intext citations could use some fine tuning on spaces, e.g. to remove trailing spaces after words, or to move superscripts to the right of punctuation, or to adjust spaces after some citations.

4. Changing the bibliography style for each entry amounts to changing a variable for the bibliography. We have to modify a function to change the intext citation style, e.g. to brackets, or (author year).

5. I stuck with only cite links here, and only articles and books. It would not get a citenum format correct, e.g. it should not be superscripted

in this case, or a citeauthor format correct. That would require some code in the replacement section that knows how to replace different types of citations.

The org-ref-unsrt-html-processor function could be broken up more, and could take some parameters to fine-tune some of these things, and generalize some things like getting the citation elements for the buffer. Overall, I think this shows that citations in org-mode with org-ref are actually pretty flexible. It is not as good as bibtex/LaTeX, and won't be for an unforseeably long time unless someone really needs high quality citations in a format other than LaTeX.

References

[1] John R. Kitchin. Examples of effective data sharing in scientific publishing. *ACS Catalysis*, 5(6):3894–3899, 2015.

[2] John R. Kitchin. Data sharing in surface science. *Surface Science*, (0):–, 2015.

[3] Zhongnan Xu and John R Kitchin. Tuning oxide activity through modification of the crystal and electronic structure: From strain to potential polymorphs. *Phys. Chem. Chem. Phys.*, 17:28943–28949, 2015.

[4] Prateek Mehta, Paul A. Salvador, and John R. Kitchin. Identifying potential bo2 oxide polymorphs for epitaxial growth candidates. *ACS Appl. Mater. Interfaces*, 6(5):3630–3639, 2014.

[5] Matthew T. Curnan and John R. Kitchin. Effects of concentration, crystal structure, magnetism, and electronic structure method on first-principles oxygen vacancy formation energy trends in perovskites. *The Journal of Physical Chemistry C*, 118(49):28776–28790, 2014.

[6] Alexander P. Hallenbeck and John R. Kitchin. Effects of o2 and so2 on the capture capacity of a primary-amine based polymeric co2 sorbent. *Industrial & Engineering Chemistry Research*, 52(31):10788–10794, 2013.

[7] Spencer D. Miller, Vladimir V. Pushkarev, Andrew J. Gellman, and John R. Kitchin. Simulating temperature programmed desorption of oxygen on pt(111) using dft derived coverage dependent desorption barriers. *Topics in Catalysis*, 57(1-4):106–117, 2014.

[8] Jacob R. Boes, Gamze Gumuslu, James B. Miller, Andrew J. Gellman, and John R. Kitchin. Estimating bulk-composition-dependent h2 adsorption energies on cuxpd1-x alloy (111) surfaces. *ACS Catalysis*, 5:1020–1026, 2015.

1.1 The working code

Here is a funtion to process the org file prior parsing during the export process. This function goes into org-export-before-parsing-hook, and takes one argument, the backend. We simply replace all the citation links with formatted HTML snippets or blocks. If the snippets get longer than a line, it will break.

We use org-ref-reftex-format-citation to generate the bibliography, which uses reftex to format a string with escape characters in it.

```
(setq org-ref-bibliography-entry-format
      '(("article" . "<li><a id=\"\%k\">%a, %t, <i>%j</i>, <b>%v(%n)</b>, %p (%y)</a>. <a href=\"%U\">link</a>. <a
        ("book" . "<li><a id=\"\%k\">%a, %t, %u (%y).</a></li>")))

(defun org-ref-unsrt-latex-processor () nil)
(defun org-ref-unsrt-html-processor ()
  "Citation processor function for the unsrt style with html output."
  (let (links
        unique-keys numbered-keys
        replacements
        bibliography-link
        bibliographystyle-link
        bibliography)
    ;; step 1 - get the citation links
    (setq links (loop for link in (org-element-map
                                      (org-element-parse-buffer) 'link 'identity)
                      if (-contains?
                          org-ref-cite-types
                          (org-element-property :type link))
                      collect link))

    ;; list of unique numbered keys. '((key number))
    (setq unique-keys (loop for i from 1
                            for key in (org-ref-get-bibtex-keys)
                            collect (list key (number-to-string i))))


    ;; (start end replacement-text)
    (setq replacements
          (loop for link in links
                collect
                (let ((path (org-element-property :path link)))
                  (loop for (key number) in unique-keys
                        do
```

```
                        (setq
                         path
                         (replace-regexp-in-string
                          key (format "<a href=\"#%s\">%s</a>" key number)
                          path)))
                  (list (org-element-property :begin link)
                        (org-element-property :end link)
                        (format "@@html:<sup>%s</sup>@@" path)))))

    ;; construct the bibliography string
    (setq bibliography
          (concat "#+begin_html
<h1>Bibliography</h1><ol>"
                  (mapconcat
                   'identity
                   (loop for (key number) in unique-keys
                         collect
                         (let* ((result (org-ref-get-bibtex-key-and-file key))
                                (bibfile (cdr result))
                                (entry (save-excursion
                                         (with-temp-buffer
                                           (insert-file-contents bibfile)
                                           (bibtex-set-dialect
                                            (parsebib-find-bibtex-dialect) t)
                                           (bibtex-search-entry key)
                                           (bibtex-parse-entry t)))))
                           ;; remove escaped & in the strings
                           (replace-regexp-in-string "\\\\&" "&"
                                                     (org-ref-reftex-format-citation
                                                      entry
                                                      (cdr (assoc (cdr (assoc "=type=" entry))
                                                                  org-ref-bibliography-entry-format))))))
                   "")
                  "</ol>
#+end_html"))

    ;; now, we need to replace each citation. We do that in reverse order so the
    ;; positions do not change.
    (loop for (start end replacement) in (reverse replacements)
          do
          (setf (buffer-substring start end) replacement))

    ;; Eliminate bibliography style links
    (loop for link in (org-element-map
                          (org-element-parse-buffer) 'link 'identity)
          if (string= "bibliographystyle"
                      (org-element-property :type link))
          do
          (setf (buffer-substring (org-element-property :begin link)
                                  (org-element-property :end link))
                ""))

    ;; replace the bibliography link with the bibliography text
    (setq bibliography-link (loop for link in (org-element-map
                                                  (org-element-parse-buffer) 'link 'identity)
                                  if (string= "bibliography"
```

```
                                                (org-element-property :type link))
                                      collect link))
    (if (> (length bibliography-link) 1)
        (error "Only one bibliography link allowed"))

    (setq bibliography-link (car bibliography-link))
    (setf (buffer-substring (org-element-property :begin bibliography-link)
                            (org-element-property :end bibliography-link))
          bibliography)))


(defun org-ref-citation-processor (backend)
  "Figure out what to call and call it"
  (let (bibliographystyle)
    (setq
     bibliographystyle
     (org-element-property
      :path (car
             (loop for link in
                   (org-element-map
                       (org-element-parse-buffer) 'link 'identity)
                   if (string= "bibliographystyle"
                               (org-element-property :type link))
                   collect link))))
    (funcall (intern (format "org-ref-%s-%s-processor" bibliographystyle backend)))))

(add-hook 'org-export-before-parsing-hook 'org-ref-citation-processor)

(browse-url (org-html-export-to-html))
```

```
#<process open ./blog.html>
```

1.2 Long appendix illustrating how we got the code to work

The first thing we need is a list of all the citation links, in the order cited. Here they are.

```
(mapcar
 (lambda (link) (org-element-property :path link))
 (loop for link in (org-element-map (org-element-parse-buffer) 'link 'identity)
       if (-contains? org-ref-cite-types (org-element-property :type link))
       collect link))
```

armiento-2014-high daza-2014-carbon-dioxid,mehta-2014-ident-poten,suntivich-2014-estim-hybrid

Now, we need to compute replacements for each citation link, and construct the bibliography. We will make a numbered, unsorted bibliography, and we want to replace each citation with the corresponding numbers, hyperlinked to the entry.

We start with a list of the keys in the order cited, and a number we will use for each one.

```
(loop for i from 1
      for key in (org-ref-get-bibtex-keys)
      collect (list key i))
```

armiento-2014-high	1
daza-2014-carbon-dioxid	2
mehta-2014-ident-poten	3
suntivich-2014-estim-hybrid	4
alesi-2012-evaluat-primar	5
day-1995-scien-englis	6

Now, we need to compute replacements for each cite link. This will be replacing each key with the number above. We will return a list of ((start end) . "replacement text") that we can use to replace each link. For fun, we make these superscripted html.

```
(let ((links (loop for link in (org-element-map (org-element-parse-buffer) 'link 'identity)
                   if (-contains? org-ref-cite-types (org-element-property :type link))
                   collect link))
      (replacements (loop for i from 1
                          for key in (org-ref-get-bibtex-keys)
                          collect (list key (number-to-string i)))))
  (loop for link in links
        collect (let ((path (org-element-property :path link)))
                  (dolist (repl replacements)
                    (setq path (replace-regexp-in-string (car repl) (nth 1 repl) path)))
                  (list (org-element-property :begin link)
                        (org-element-property :end link)
                        (format "<sup>%s</sup>" path)))))
```

604	628	¹
643	723	^{2,3,4}
803	826	¹
924	954	⁵
996	1022	⁶
1062	1085	¹

We also need to compute the bibliography for each key. We will use org-ref-reftex-format-citation to do this. For that we need the parsed bibtex entries, and a format string. org-ref provides most of this.

```
(setq org-ref-bibliography-entry-format
      '(("article" . "<li>%a, %t, <i>%j</i>, <b>%v(%n)</b>, %p (%y). <a href=\"%U\">link</a>. <a href=\"http://dx.d
        ("book" . "<li>%a, %t, %u (%y).</li>")))

(concat "<h1>Bibliography</h1><br><ol>"
        (mapconcat
         'identity
         (loop for key in (org-ref-get-bibtex-keys)
               collect
               (let* ((result (org-ref-get-bibtex-key-and-file key))
                      (bibfile (cdr result))
                      (entry (save-excursion
                               (with-temp-buffer
                                 (insert-file-contents bibfile)
                                 (bibtex-set-dialect (parsebib-find-bibtex-dialect) t)
                                 (bibtex-search-entry key)
                                 (bibtex-parse-entry)))))
                 (org-ref-reftex-format-citation
                  entry
                  (cdr (assoc (cdr (assoc "=type=" entry))
                              org-ref-bibliography-entry-format)))))
         "")
        "</ol>")
```