

Scribib: Extra Scribble Libraries

Version 4.2.5

April 2, 2010

Contents

1 Writing Examples with Pict Results	3
2 Figures	4
3 Bibliographies	6

1 Writing Examples with Pict Results

```
(require scriblib/gui-eval)
```

The `scriblib/gui-eval` library support example evaluations with results that are `slideshow` picts.

The trick is that `scheme/gui` is not generally available when rendering documentation, because it requires a GUI context. The picture output is rendered to an image file when the `MREVAL` environment variable is set, so run the enclosing document once with the environment variable to generate the images. Future runs (with the environment variable unset) use the generated image.

```
(gui-interaction datum ...)  
(gui-interaction-eval datum ...)  
(gui-interaction-eval-show datum ...)  
(gui-schemeblock+eval datum ...)  
(gui-schememod+eval datum ...)  
(gui-def+int datum ...)  
(gui-defs+int datum ...)
```

Like `interaction`, etc., but actually evaluating the forms only when the `MREVAL` environment variable is set, and then in an evaluator that is initialized with `scheme/gui/base` and `slideshow`.

2 Figures

(require scriblib/figure)

(figure-style-extras) → list?

Include the content of the result list in the style of a document part that includes all figures. These style extras pull in HTML and Latex rendering support.

(figure tag caption pre-flow ...) → block?

tag : string?
caption : content?
pre-flow : pre-flow?

(figure* tag caption pre-flow ...) → block?

tag : string?
caption : content?
pre-flow : pre-flow?

(figure** tag caption pre-flow ...) → block?

tag : string?
caption : content?
pre-flow : pre-flow?

Creates a figure. The given *tag* is for use with `figure-ref` or `fFigure-ref`. The *caption* is an element. The *pre-flow* is decoded as a flow.

For HTML output, the `figure*` and `figure*` functions center the figure content, while `figure**` allows the content to be wider than the document body.

For two-column latex output, `figure*` and `figure**` generate a figure that spans columns.

(figure-ref tag) → element?

tag : string?

Generates a reference to a figure, using a lowercase word “figure”.

(Figure-ref tag) → element?

tag : string?

Generates a reference to a figure, capitalizing the word “Figure”.

(Figure-target tag) → element?

tag : string?

Generates a new figure label. This function is normally not used directly, since it is used by [figure](#).

3 Bibliographies

```
(require scriblib/autobib)
```

```
(autobib-style-extras) → list?
```

Includes the content of the result list in the style of a document part that includes all figures. These style extras pull in HTML and Latex rendering support.

```
(define-cite ~cite-id citet-id generate-bibliography-id)
```

Binds `~cite-id`, `citet-id`, and `generate-bibliography-id`, which share state to accumulate and render citations.

The function bound to `~cite-id` produces a citation with a preceding non-breaking space. It has the contract

```
((bib?) () (listof bib?) . ->* . element?)
```

The function bound to `citet-id` has the same contract as the function for `~cite-id`, but it generates an element suitable for use as a noun referring to the document or its author.

The function bound to `generate-bibliography-id` generates the section for the bibliography. It has the contract

```
((() (#:tag [tag "doc-bibliography"]) null? . ->* . part?)
```

```
(bib? v) → boolean?
```

```
  v : any/c
```

Returns `#t` if `v` is a value produced by `make-bib` or `in-bib`, `#f` otherwise.

```
(make-bib #:title title
          #:author author
          #:is-book? is-book?
          #:location location
          #:date date
          #:url url) → bib?
```

```
title : any/c
author : any/c = #f
is-book? : any/c = #f
location : any/c = #f
date : any/c = #f
```

```
url : string? = #f
```

Produces a value that represents a document to cite. Except for *is-book?* and *url*, the arguments are used as elements, except that *#f* means that the information is not supplied. Functions like *proceedings-location*, *author-name*, and *authors* help produce elements in a standard format.

An element produced by a function like *author-name* tracks first, last names, and name suffixes separately, so that names can be ordered and rendered correctly. When a string is provided as an author name, the last non-empty sequence of ASCII alphabetic characters after a space is treated as the author name, and the rest is treated as the first name.

```
(in-bib orig where) → bib?  
  orig : bib?  
  where : string?
```

Extends a bib value so that the rendered citation is suffixed with *where*, which might be a page or chapter number.

```
(proceedings-location location  
  [#:pages pages  
   #:series series  
   #:volume volume]) → element?  
  
location : any/c  
pages : (or (list/c any/c any/c) #f) = #f  
series : any/c = #f  
volume : any/c = #f
```

Combines elements to generate an element that is suitable for describing a paper's location within a conference or workshop proceedings.

```
(journal-location title  
  [#:pages pages  
   #:number number  
   #:volume volume]) → element?  
  
title : any/c  
pages : (or (list/c any/c any/c) #f) = #f  
number : any/c = #f  
volume : any/c = #f
```

Combines elements to generate an element that is suitable for describing a paper's location within a journal.

```
(book-location [#:edition edition
                #:publisher publisher]) → element?
  edition : any/c = #f
  publisher : any/c = #f
```

Combines elements to generate an element that is suitable for describing a book's location.

```
(techrpt-location [#:institution institution]
                  #:number number) → element?
  institution : edition = any/c
  number : any/c
```

Combines elements to generate an element that is suitable for describing a technical report's location.

```
(dissertation-location [#:institution institution
                        #:number degree]) → element?
  institution : edition = any/c
  degree : any/c = "PhD"
```

Combines elements to generate an element that is suitable for describing a dissertation.

```
(author-name first last [#:suffix suffix]) → element?
  first : any/c
  last : any/c
  suffix : any/c = #f
```

Combines elements to generate an element that is suitable for describing an author's name, especially where the last name is not merely a sequence of ASCII alphabet letters or where the name has a suffix (such as "Jr.").

```
(authors name ...) → element?
  name : any/c
```

Combines multiple author elements into one, so that it is rendered and alphabetized appropriately. If a *name* is a string, it is parsed in the same way as by `make-bib`.

```
(org-author-name name) → element?
  name : any/c
```

Converts an element for an organization name to one suitable for use as a bib-value author.

`(other-authors)` → `element?`

Generates an element that is suitable for use as a “others” author. When combined with another author element via `authors`, the one created by `other-authors` renders as “et al.”

`(editor name)` → `element?`
`name` : `name/c`

Takes an author-name element and create one that represents the editor of a collection. If a `name` is a string, it is parsed in the same way as by `make-bib`.