HTML: Parsing Library

Version 4.2.5

April 2, 2010

```
(require html)
```

The html library provides functions to read html documents and structures to represent them.

```
(read-xhtml port) → html?
  port : input-port?
(read-html port) → html?
  port : input-port?
```

Reads (X)HTML from a port, producing an html instance.

```
(read-html-as-xml port) → (listof content/c)
  port : input-port?
```

Reads HTML from a port, producing an X-expression compatible with the xml library (which defines content/c).

```
(read-html-comments) → boolean?
(read-html-comments v) → void?
  v : any/c
```

If v is not #f, then comments are read and returned. Defaults to #f.

```
(use-html-spec) → boolean?
(use-html-spec v) → void?
  v : any/c
```

If v is not #f, then the HTML must respect the HTML specification with regards to what elements are allowed to be the children of other elements. For example, the top-level "<html>"

element may only contain a `"<body>"` and `"<head>"` element. Defaults to `#f`.

1 Example

```
(module html-example scheme

  ; Some of the symbols in html and xml conflict with
  ; each other and with scheme/base language, so we prefix
  ; to avoid namespace conflict.
  (require (prefix-in h: html)
           (prefix-in x: xml))

  (define an-html
    (h:read-xhtml
     (open-input-string
      (string-append
       "<html><head><title>My title</title></head><body>"
       "<p>Hello world</p><p><b>Testing</b>!</p>"
       "</body></html>"))))

  ; extract-pcdata: html-content -> (listof string)
  ; Pulls out the pcdata strings from some-content.
  (define (extract-pcdata some-content)
    (cond [(x:pcdata? some-content)
           (list (x:pcdata-string some-content))]
          [(x:entity? some-content)
           (list)]
          [else
           (extract-pcdata-from-element some-content)]))

  ; extract-pcdata-from-element: html-element -> (listof string)
  ; Pulls out the pcdata strings from an-html-element.
  (define (extract-pcdata-from-element an-html-element)
    (match an-html-element
      [(struct h:html-full (attributes content))
       (apply append (map extract-pcdata content))]

      [(struct h:html-element (attributes))
       '()]))

  (printf "~s~n" (extract-pcdata an-html)))

> (require 'html-example)
("My title" "Hello world" "Testing" "!")
```

2 HTML Structures

pcdata, entity, and attribute are defined in the xml documentation.

A html-content is either

- html-element
- pcdata
- entity

```
(struct html-element (attributes))
  attributes : (listof attribute)
```

Any of the structures below inherits from html-element.

```
(struct (html-full struct:html-element) (content))
  content : (listof html-content)
```

Any html tag that may include content also inherits from html-full without adding any additional fields.

```
(struct (html html-full) ())
```

A html is (make-html (listof attribute) (listof Contents-of-html))

A Contents-of-html is either

- body
- head

```
(struct (div html-full) ())
```

A div is (make-div (listof attribute) (listof G2))

```
(struct (center html-full) ())
```

A center is (make-center (listof attribute) (listof G2))

```
(struct (blockquote html-full) ())
```

A blockquote is (make-blockquote (listof attribute) G2)

(struct (ins html-full) ())

An Ins is (make-ins (listof attribute) (listof G2))

(struct (del html-full) ())

A del is (make-del (listof attribute) (listof G2))

(struct (dd html-full) ())

A dd is (make-dd (listof attribute) (listof G2))

(struct (li html-full) ())

A li is (make-li (listof attribute) (listof G2))

(struct (th html-full) ())

A th is (make-th (listof attribute) (listof G2))

(struct (td html-full) ())

A td is (make-td (listof attribute) (listof G2))

(struct (iframe html-full) ())

An iframe is (make-iframe (listof attribute) (listof G2))

(struct (noframes html-full) ())

A noframes is (make-noframes (listof attribute) (listof G2))

(struct (noscript html-full) ())

A noscript is (make-noscript (listof attribute) (listof G2))

(struct (style html-full) ())

A style is (make-style (listof attribute) (listof pcdata))

```
(struct (script html-full) ())
```

A `script` is `(make-script (listof attribute) (listof pcdata))`

```
(struct (basefont html-element) ())
```

A `basefont` is `(make-basefont (listof attribute))`

```
(struct (br html-element) ())
```

A `br` is `(make-br (listof attribute))`

```
(struct (area html-element) ())
```

An `area` is `(make-area (listof attribute))`

```
(struct (alink html-element) ())
```

A `alink` is `(make-alink (listof attribute))`

```
(struct (img html-element) ())
```

An `img` is `(make-img (listof attribute))`

```
(struct (param html-element) ())
```

A `param` is `(make-param (listof attribute))`

```
(struct (hr html-element) ())
```

A `hr` is `(make-hr (listof attribute))`

```
(struct (input html-element) ())
```

An `input` is `(make-input (listof attribute))`

```
(struct (col html-element) ())
```

A `col` is `(make-col (listof attribute))`

```
(struct (isindex html-element) ())
```

An isindex is (make-isindex (listof attribute))

```
(struct (base html-element) ())
```

A base is (make-base (listof attribute))

```
(struct (meta html-element) ())
```

A meta is (make-meta (listof attribute))

```
(struct (option html-full) ())
```

An option is (make-option (listof attribute) (listof pcdata))

```
(struct (textarea html-full) ())
```

A textarea is (make-textarea (listof attribute) (listof pcdata))

```
(struct (title html-full) ())
```

A title is (make-title (listof attribute) (listof pcdata))

```
(struct (head html-full) ())
```

A head is (make-head (listof attribute) (listof Contents-of-head))

A Contents-of-head is either

- base
- isindex
- alink
- meta
- object
- script
- style

- title

(struct (tr html-full) ())

A tr is (make-tr (listof attribute) (listof Contents-of-tr))

A Contents-of-tr is either

- td
- th

(struct (colgroup html-full) ())

A colgroup is (make-colgroup (listof attribute) (listof col))

(struct (thead html-full) ())

A thead is (make-thead (listof attribute) (listof tr))

(struct (tfoot html-full) ())

A tfoot is (make-tfoot (listof attribute) (listof tr))

(struct (tbody html-full) ())

A tbody is (make-tbody (listof attribute) (listof tr))

(struct (tt html-full) ())

A tt is (make-tt (listof attribute) (listof G5))

(struct (i html-full) ())

An i is (make-i (listof attribute) (listof G5))

(struct (b html-full) ())

A b is (make-b (listof attribute) (listof G5))

(struct (u html-full) ())

An u is (make-u (listof attribute) (listof G5))

(struct (s html-full) ())

A s is (make-s (listof attribute) (listof G5))

(struct (strike html-full) ())

A strike is (make-strike (listof attribute) (listof G5))

(struct (big html-full) ())

A big is (make-big (listof attribute) (listof G5))

(struct (small html-full) ())

A small is (make-small (listof attribute) (listof G5))

(struct (em html-full) ())

An em is (make-em (listof attribute) (listof G5))

(struct (strong html-full) ())

A strong is (make-strong (listof attribute) (listof G5))

(struct (dfn html-full) ())

A dfn is (make-dfn (listof attribute) (listof G5))

(struct (code html-full) ())

A code is (make-code (listof attribute) (listof G5))

(struct (samp html-full) ())

A samp is (make-samp (listof attribute) (listof G5))

(struct (kbd html-full) ())

A kbd is (make-kbd (listof attribute) (listof G5))

```
(struct (var html-full) ())
```

A var is (make-var (listof attribute) (listof G5))

```
(struct (cite html-full) ())
```

A cite is (make-cite (listof attribute) (listof G5))

```
(struct (abbr html-full) ())
```

An abbr is (make-abbr (listof attribute) (listof G5))

```
(struct (acronym html-full) ())
```

An acronym is (make-acronym (listof attribute) (listof G5))

```
(struct (sub html-full) ())
```

A sub is (make-sub (listof attribute) (listof G5))

```
(struct (sup html-full) ())
```

A sup is (make-sup (listof attribute) (listof G5))

```
(struct (span html-full) ())
```

A span is (make-span (listof attribute) (listof G5))

```
(struct (bdo html-full) ())
```

A bdo is (make-bdo (listof attribute) (listof G5))

```
(struct (font html-full) ())
```

A font is (make-font (listof attribute) (listof G5))

```
(struct (p html-full) ())
```

A p is (make-p (listof attribute) (listof G5))

```
(struct (h1 html-full) ())
```

A h1 is (make-h1 (listof attribute) (listof G5))

```
(struct (h2 html-full) ())
```

A h2 is (make-h2 (listof attribute) (listof G5))

```
(struct (h3 html-full) ())
```

A h3 is (make-h3 (listof attribute) (listof G5))

```
(struct (h4 html-full) ())
```

A h4 is (make-h4 (listof attribute) (listof G5))

```
(struct (h5 html-full) ())
```

A h5 is (make-h5 (listof attribute) (listof G5))

```
(struct (h6 html-full) ())
```

A h6 is (make-h6 (listof attribute) (listof G5))

```
(struct (q html-full) ())
```

A q is (make-q (listof attribute) (listof G5))

```
(struct (dt html-full) ())
```

A dt is (make-dt (listof attribute) (listof G5))

```
(struct (legend html-full) ())
```

A legend is (make-legend (listof attribute) (listof G5))

```
(struct (caption html-full) ())
```

A caption is (make-caption (listof attribute) (listof G5))

`(struct (table html-full) ())`

A `table` is `(make-table (listof attribute) (listof Contents-of-table))`

A `Contents-of-table` is either

- `caption`
- `col`
- `colgroup`
- `tbody`
- `tfoot`
- `thead`

`(struct (button html-full) ())`

A `button` is `(make-button (listof attribute) (listof G4))`

`(struct (fieldset html-full) ())`

A `fieldset` is `(make-fieldset (listof attribute) (listof Contents-of-fieldset))`

A `Contents-of-fieldset` is either

- `legend`
- G2

`(struct (optgroup html-full) ())`

An `optgroup` is `(make-optgroup (listof attribute) (listof option))`

`(struct (select html-full) ())`

A `select` is `(make-select (listof attribute) (listof Contents-of-select))`

A `Contents-of-select` is either

- `optgroup`

- option

(struct (label html-full) ())

A label is (make-label (listof attribute) (listof G6))

(struct (form html-full) ())

A form is (make-form (listof attribute) (listof G3))

(struct (ol html-full) ())

An ol is (make-ol (listof attribute) (listof li))

(struct (ul html-full) ())

An ul is (make-ul (listof attribute) (listof li))

(struct (dir html-full) ())

A dir is (make-dir (listof attribute) (listof li))

(struct (menu html-full) ())

A menu is (make-menu (listof attribute) (listof li))

(struct (dl html-full) ())

A dl is (make-dl (listof attribute) (listof Contents-of-dl))

A Contents-of-dl is either

- dd
- dt

(struct (pre html-full) ())

A pre is (make-pre (listof attribute) (listof Contents-of-pre))

A Contents-of-pre is either

- G9
- G11

`(struct (object html-full) ())`

An `object` is `(make-object (listof attribute) (listof Contents-of-object-applet))`

`(struct (applet html-full) ())`

An `applet` is `(make-applet (listof attribute) (listof Contents-of-object-applet))`

A `Contents-of-object-applet` is either

- `param`
- G2

`(struct (map html-full) ())`

A Map is `(make-map (listof attribute) (listof Contents-of-map))`

A `Contents-of-map` is either

- `area`
- `fieldset`
- `form`
- `isindex`
- G10

`(struct (a html-full) ())`

An `a` is `(make-a (listof attribute) (listof Contents-of-a))`

A `Contents-of-a` is either

- `label`

- G7

`(struct (address html-full) ())`

An `address` is `(make-address (listof attribute) (listof Contents-of-address))`

A `Contents-of-address` is either

- `p`
- G5

`(struct (body html-full) ())`

A `body` is `(make-body (listof attribute) (listof Contents-of-body))`

A `Contents-of-body` is either

- `del`
- `ins`
- G2

A `G12` is either

- `button`
- `iframe`
- `input`
- `select`
- `textarea`

A `G11` is either

- `a`
- `label`
- `G12`

A G10 is either

- address
- blockquote
- center
- dir
- div
- dl
- h1
- h2
- h3
- h4
- h5
- h6
- hr
- menu
- noframes
- noscript
- ol
- p
- pre
- table
- ul

A G9 is either

- abbr
- acronym
- b

- bdo
- br
- cite
- code
- dfn
- em
- i
- kbd
- map
- pcdata
- q
- s
- samp
- script
- span
- strike
- strong
- tt
- u
- var

A G8 is either

- applet
- basefont
- big
- font
- img
- object

- small
- sub
- sup
- G9

A G7 is either

- G8
- G12

A G6 is either

- a
- G7

A G5 is either

- label
- G6

A G4 is either

- G8
- G10

A G3 is either

- fieldset
- isindex
- G4
- G11

A G2 is either

- form
- G3