macro-html Manual

Max Rottenkolber

Sunday, 26 July 2015

Table of Contents

1 Abstract **1**

2 Usage **1**
2.1 List of defined regular element macros 2
2.2 List of defined single element macros 2

3 Dependencies **3**

1 Abstract

`macro-html` is a DSL for generating HTML[1]. It aims to be fast, modular, cachable and concise. It does so by defining each element as a macro which expands to code printing the respective HTML source. It also employs a readtable syntax for element attributes.

`macro-html.widgets` is a collection of widget functions. Those functions utilize `macro-html` to print compound HTML elements such as lists, forms or even complete documents. The provided widgets are documented in *macro-html Widgets* (`widgets.html`).

- 1. *HTML5* (`http://www.w3.org/TR/html5/`)

2 Usage

`macro-html` exports an *element macro* for every valid HTML element as of HTML5. An element macro prints the respective HTML element including its attributes, child elements and text nodes to `*standard-output*`. Single element macros such as `br` take a *property list* of attributes as their arguments, e.g. `(br :class "foo" ...)`. Regular tag macros such as `p` take an arbitrary number of children as arguments. The first argument

can optionally be an attribute list, e.g. `(p '(:attributes :class "bar" ...) "foo" ...)`.

In order to ease specifying attributes for regular element macros the readtable `macro-html:syntax` provides a specialized reader syntax, e.g. `(p [:class "bar" ...] "foo" ...)`.

Every child form gets evaluated exactly once. If a form evaluates to a string or pathname it will be escaped and printed inside the element, otherwise its return value will be ignored. Element macros can be nested to produce compound HTML output.

```
(p [:class "foo" :id "bar"]
   "Hello, " (b "World") "!"
   (br :class "baz")
   (symbol-name 'list))
▷ <P CLASS="foo" ID="bar">Hello, <B>World</B>
▷ <BR CLASS="baz">LIST</P>
```

Example usage of `macro-html`.

The function `html-doctype` prints the HTML `!DOCTYPE` element to `*standard-output*`. The macro `text` evaluates its body forms as if they were children of a element macro and can be used to explicitly print text nodes.

```
(defun hello (name)
  (text "Hello " name "!"))

(h1 (hello "Joe"))
▷ <H1>Hello Joe!</H1>
```

Example usage of `text`.

2.1 List of defined regular element macros

`a`, `abbr`, `address`, `article`, `aside`, `audio`, `b`, `bdi`, `bdo`, `blockquote`, `body`, `button`, `canvas`, `caption`, `cite`, `code`, `colgroup`, `datalist`, `dd`, `del`, `details`, `dfn`, `div`, `dl`, `dt`, `em`, `embed`, `fieldset`, `figcaption`, `figure`, `footer`, `form`, `h1`, `h2`, `h3`, `h4`, `h5`, `h6`, `head`, `hgroup`, `header`, `html`, `i`, `iframe`, `ins`, `keygen`, `kbd`, `label`, `legend`, `li`, `map`, `mark`, `menu`, `meter`, `nav`, `noscript`, `object`, `ol`, `optgroup`, `option`, `output`, `p`, `pre`, `progress`, `q`, `rp`, `rt`, `ruby`, `s`, `samp`, `script`, `section`, `select`, `small`, `span`, `strong`, `style`, `sub`, `sup`, `table`, `tbody`, `td`, `textarea`, `tfoot`, `th`, `thead`, `time`, `title`, `tr`, `u`, `ul`, `var`, `video`, `wbr`

2.2 List of defined single element macros

`area`, `base`, `br`, `col`, `command`, `hr`, `img`, `input`, `link`, `meta`, `option`, `param`, `source`, `track`

3 Dependencies

`macro-html` depends on `named-readtables` and shadows `map` and `time` as those symbols collide with HTML element names.