NAME

HTML::Tagset − data tables useful in parsing HTML

VERSION

Version 3.20

SYNOPSIS

```
use HTML::Tagset;
# Then use any of the items in the HTML::Tagset package
#  as need arises
```

DESCRIPTION

This module contains several data tables useful in various kinds of HTML parsing operations.

Note that all tag names used are lowercase.

In the following documentation, a ''hashset'' is a hash being used as a set — the hash conveys that its keys are there, and the actual values associated with the keys are not significant. (But what values are there, are always true.)

VARIABLES

Note that none of these variables are exported.

hashset `%HTML::Tagset::emptyElement`

This hashset has as values the tag-names (GIs) of elements that cannot have content. (For example, ''base'', ''br'', ''hr''.) So`$HTML::Tagset::emptyElement{'hr'}` e xists and is true. `$HTML::Tagset::emptyElement{'dl'}` does not exist, and so is not true.

hashset `%HTML::Tagset::optionalEndTag`

This hashset lists tag-names for elements that can have content, but whose end-tags are generally, ''safely'', omissible. Example:`$HTML::Tagset::emptyElement{'li'}` e xists and is true.

hash `%HTML::Tagset::linkElements`

Values in this hash are tagnames for elements that might contain links, and the value for each is a reference to an array of the names of attributes whose values can be links.

hash `%HTML::Tagset::boolean_attr`

This hash (not hashset) lists what attributes of what elements can be printed without showing the value (for example, the ''noshade'' attribute of ''hr'' elements). For elements with only one such attribute, its value is simply that attribute name. For elements with many such attributes, the value is a reference to a hashset containing all such attributes.

hashset `%HTML::Tagset::isPhraseMarkup`

This hashset contains all phrasal-level elements.

hashset `%HTML::Tagset::is_Possible_Strict_P_Content`

This hashset contains all phrasal-level elements that be content of a P element, for a strict model of HTML.

hashset `%HTML::Tagset::isHeadElement`

This hashset contains all elements that elements that should be present only in the 'head' element of an HTML document.

hashset `%HTML::Tagset::isList`

This hashset contains all elements that can contain ''li'' elements.

hashset `%HTML::Tagset::isTableElement`

This hashset contains all elements that are to be found only in/under a ''table'' element.

hashset `%HTML::Tagset::isFormElement`

This hashset contains all elements that are to be found only in/under a ''form'' element.

hashset `%HTML::Tagset::isBodyElement`

This hashset contains all elements that are to be found only in/under the ''body'' element of an HTML document.

hashset `%HTML::Tagset::isHeadOrBodyElement`
This hashset includes all elements that I notice can fall either in the head or in the body.

hashset `%HTML::Tagset::isKnown`
This hashset lists all known HTML elements.

hashset `%HTML::Tagset::canTighten`
This hashset lists elements that might have ignorable whitespace as children or siblings.

array `@HTML::Tagset::p_closure_barriers`
This array has a meaning that I have only seen a need for in `HTML::TreeBuilder`, but I include it here on the off chance that someone might find it of use:

When we see a ''<p>'' token, we go lookup up the lineage for a p element we might have to minimize. At first sight, we might say that if there's a p anywhere in the lineage of this new p, it should be closed. But that's wrong. Consider this document:

```
<html>
  <head>
    <title>foo</title>
  </head>
  <body>
    <p>foo
      <table>
        <tr>
          <td>
             foo
             <p>bar
          </td>
        </tr>
      </table>
    </p>
  </body>
</html>
```

The second p is quite legally inside a much higher p.

My formalization of the reason why this is legal, but this:

```
<p>foo<p>bar</p></p>
```

isn't, is that something about the table constitutes a ''barrier'' to the application of the rule about what p must minimize.

So `@HTML::Tagset::p_closure_barriers` is the list of all such barrier-tags.

hashset `%isCDATA_Parent`
This hashset includes all elements whose content is CDATA.

CAVEATS

You may find it useful to alter the behavior of modules (like `HTML::Element` or `HTML::TreeBuilder`) that use `HTML::Tagset`'s data tables by altering the data tables themselves. You are welcome to try, but be careful; and be aware that different modules may or may react differently to the data tables being changed.

Note that it may be inappropriate to use these tables for *producing* HTML — for example, `%isHeadOrBodyElement` lists the tagnames for all elements that can appear either in the head or in the body, such as ''script''. That doesn't mean that I am saying your code that produces HTML should feel free to put script elements in either place! If you are producing programs that spit out HTML, you should be *intimately* familiar with the DTDs for HTML or XHTML (available at `http://www.w3.org/`), and you should slavishly obey them, not the data tables in this document.

SEE ALSO

HTML::Element, HTML::TreeBuilder, HTML::LinkExtor

COPYRIGHT & LICENSE

ACKNOWLEDGEMENTS

Most of the code/data in this module was adapted from code written by Gisle Aas for `HTML::Element`, `HTML::TreeBuilder`, and `HTML::LinkExtor`. Then it was maintained by Sean M. Burke.

AUTHOR

Current maintainer: Andy Lester, `<andy at petdance.com>`

BUGS

Please report any bugs or feature requests to `bug−html−tagset at rt.cpan.org`, or through the web interface at <http://rt.cpan.org/NoAuth/ReportBug.html?Queue=HTML−Tagset>. I will be notified, and then you'll automatically be notified of progress on your bug as I make changes.