html5lib Documentation

Release 0.9999999999-dev

James Graham, Geoffrey Sneddon, and contributors

Jul 23, 2017

Contents

html5lib is a pure-python library for parsing HTML. It is designed to conform to the WHATWG HTML specification, as is implemented by all major web browsers.

CHAPTER 1

Usage

Simple usage follows this pattern:

```
import html5lib
with open("mydocument.html", "rb") as f:
    document = html5lib.parse(f)
```

or:

```
import html5lib
document = html5lib.parse("<p>Hello World!")
```

By default, the `document` will be an `xml.etree` element instance. Whenever possible, html5lib chooses the accelerated `ElementTree` implementation (i.e. `xml.etree.cElementTree` on Python 2.x).

Two other tree types are supported: `xml.dom.minidom` and `lxml.etree`. To use an alternative format, specify the name of a treebuilder:

```
import html5lib
with open("mydocument.html", "rb") as f:
    lxml_etree_document = html5lib.parse(f, treebuilder="lxml")
```

When using with `urllib2` (Python 2), the charset from HTTP should be pass into html5lib as follows:

```
from contextlib import closing
from urllib2 import urlopen
import html5lib

with closing(urlopen("http://example.com/")) as f:
    document = html5lib.parse(f, transport_encoding=f.info().getparam("charset"))
```

When using with `urllib.request` (Python 3), the charset from HTTP should be pass into html5lib as follows:

```
from urllib.request import urlopen
import html5lib
```

```
with urlopen("http://example.com/") as f:
    document = html5lib.parse(f, transport_encoding=f.info().get_content_charset())
```

To have more control over the parser, create a parser object explicitly. For instance, to make the parser raise exceptions on parse errors, use:

```
import html5lib
with open("mydocument.html", "rb") as f:
    parser = html5lib.HTMLParser(strict=True)
    document = parser.parse(f)
```

When you're instantiating parser objects explicitly, pass a treebuilder class as the `tree` keyword argument to use an alternative document format:

```
import html5lib
parser = html5lib.HTMLParser(tree=html5lib.getTreeBuilder("dom"))
minidom_document = parser.parse("<p>Hello World!")
```

More documentation is available at https://html5lib.readthedocs.io/.

CHAPTER 2

Installation

html5lib works on CPython 2.6+, CPython 3.3+ and PyPy. To install it, use:

```
$ pip install html5lib
```

CHAPTER 3

Optional Dependencies

The following third-party libraries may be used for additional functionality:

- `datrie` can be used under CPython to improve parsing performance (though in almost all cases the improvement is marginal);
- `lxml` is supported as a tree format (for both building and walking) under CPython (but *not* PyPy where it is known to cause segfaults);
- `genshi` has a treewalker (but not builder); and
- `chardet` can be used as a fallback when character encoding cannot be determined.

CHAPTER 4

Bugs

Please report any bugs on the issue tracker.

CHAPTER 5

Tests

Unit tests require the `pytest` and `mock` libraries and can be run using the `py.test` command in the root directory; `ordereddict` is required under Python 2.6. All should pass.

Test data are contained in a separate html5lib-tests repository and included as a submodule, thus for git checkouts they must be initialized:

```
$ git submodule init
$ git submodule update
```

If you have all compatible Python implementations available on your system, you can run tests on all of them using the `tox` utility, which can be found on PyPI.

CHAPTER 6

Questions?

There's a mailing list available for support on Google Groups, html5lib-discuss, though you may get a quicker response asking on IRC in #whatwg on irc.freenode.net.

The moving parts

html5lib consists of a number of components, which are responsible for handling its features.

Tree builders

The parser reads HTML by tokenizing the content and building a tree that the user can later access. There are three main types of trees that html5lib can build:

- `etree` - this is the default; builds a tree based on `xml.etree`, which can be found in the standard library. Whenever possible, the accelerated `ElementTree` implementation (i.e. `xml.etree.cElementTree` on Python 2.x) is used.
- `dom` - builds a tree based on `xml.dom.minidom`.
- `lxml.etree` - uses lxml's implementation of the `ElementTree` API. The performance gains are relatively small compared to using the accelerated `ElementTree` module.

You can specify the builder by name when using the shorthand API:

```
import html5lib
with open("mydocument.html", "rb") as f:
    lxml_etree_document = html5lib.parse(f, treebuilder="lxml")
```

When instantiating a parser object, you have to pass a tree builder class in the `tree` keyword attribute:

```
import html5lib
parser = html5lib.HTMLParser(tree=SomeTreeBuilder)
document = parser.parse("<p>Hello World!")
```

To get a builder class by name, use the `getTreeBuilder` function:

```
import html5lib
parser = html5lib.HTMLParser(tree=html5lib.getTreeBuilder("dom"))
minidom_document = parser.parse("<p>Hello World!")
```

The implementation of builders can be found in html5lib/treebuilders/.

Tree walkers

Once a tree is ready, you can work on it either manually, or using a tree walker, which provides a streaming view of the tree. html5lib provides walkers for all three supported types of trees (`etree`, `dom` and `lxml`).

The implementation of walkers can be found in html5lib/treewalkers/.

Walkers make consuming HTML easier. html5lib uses them to provide you with has a couple of handy tools.

HTMLSerializer

The serializer lets you write HTML back as a stream of bytes.

```
>>> import html5lib
>>> element = html5lib.parse('<p xml:lang="pl">Witam wszystkich')
>>> walker = html5lib.getTreeWalker("etree")
>>> stream = walker(element)
>>> s = html5lib.serializer.HTMLSerializer()
>>> output = s.serialize(stream)
>>> for item in output:
...   print("%r" % item)
'<p'
' '
'xml:lang'
'='
'pl'
'>'
'Witam wszystkich'
```

You can customize the serializer behaviour in a variety of ways, consult the `HTMLSerializer` documentation.

Filters

You can alter the stream content with filters provided by html5lib:

- *`alphabeticalattributes.Filter`* sorts attributes on tags to be in alphabetical order
- *`inject_meta_charset.Filter`* sets a user-specified encoding in the correct `<meta>` tag in the `<head>` section of the document
- *`lint.Filter`* raises `LintError` exceptions on invalid tag and attribute names, invalid PCDATA, etc.
- *`optionaltags.Filter`* removes tags from the stream which are not necessary to produce valid HTML
- *`sanitizer.Filter`* removes unsafe markup and CSS. Elements that are known to be safe are passed through and the rest is converted to visible text. The default configuration of the sanitizer follows the WHATWG Sanitization Rules.
- *`whitespace.Filter`* collapses all whitespace characters to single spaces unless they're in `<pre/>` or `textarea` tags.

To use a filter, simply wrap it around a stream:

```
>>> import html5lib
>>> from html5lib.filters import sanitizer
>>> dom = html5lib.parse("<p><script>alert('Boo!')", treebuilder="dom")
>>> walker = html5lib.getTreeWalker("dom")
>>> stream = walker(dom)
>>> clean_stream = sanitizer.Filter(stream)
```

Tree adapters

Used to translate one type of tree to another. More documentation pending, sorry.

Encoding discovery

Parsed trees are always Unicode. However a large variety of input encodings are supported. The encoding of the document is determined in the following way:

- The encoding may be explicitly specified by passing the name of the encoding as the encoding parameter to the *parse()* method on HTMLParser objects.
- If no encoding is specified, the parser will attempt to detect the encoding from a <meta> element in the first 512 bytes of the document (this is only a partial implementation of the current HTML 5 specification).
- If no encoding can be found and the chardet library is available, an attempt will be made to sniff the encoding from the byte pattern.
- If all else fails, the default encoding will be used. This is usually Windows-1252, which is a common fallback used by Web browsers.

Tokenizers

The part of the parser responsible for translating a raw input stream into meaningful tokens is the tokenizer. Currently html5lib provides two.

To set up a tokenizer, simply pass it when instantiating a *HTMLParser*:

```
import html5lib
from html5lib import sanitizer

p = html5lib.HTMLParser(tokenizer=sanitizer.HTMLSanitizer)
p.parse("<p>Surprise!<script>alert('Boo!');</script>")
```

HTMLTokenizer

This is the default tokenizer, the heart of html5lib. The implementation can be found in html5lib/tokenizer.py.

HTMLSanitizer

This is a tokenizer that removes unsafe markup and CSS styles from the input. Elements that are known to be safe are passed through and the rest is converted to visible text. The default configuration of the sanitizer follows the WHATWG Sanitization Rules.

The implementation can be found in html5lib/sanitizer.py.

html5lib

html5lib Package

html5lib Package

HTML parsing library based on the WHATWG "HTML5" specification. The parser is designed to be compatible with existing HTML found in the wild and implements well-defined error recovery that is largely compatible with modern desktop web browsers.

Example usage:

import html5lib f = open("my_document.html") tree = html5lib.parse(f)

class html5lib.__init__.**HTMLParser**(*tree=None*, *strict=False*, *namespaceHTMLElements=True*, *debug=False*)

Bases: object

HTML parser. Generates a tree structure from a stream of (possibly malformed) HTML

adjustForeignAttributes(*token*)

adjustMathMLAttributes(*token*)

adjustSVGAttributes(*token*)

documentEncoding
The name of the character encoding that was used to decode the input stream, or None if that is not determined yet.

isHTMLIntegrationPoint(*element*)

isMathMLTextIntegrationPoint(*element*)

mainLoop()

normalizeToken(*token*)
HTML5 specific normalizations to the token stream

normalizedTokens()

parse(*stream*, **args*, ***kwargs*)
Parse a HTML document into a well-formed tree

stream - a filelike object or string containing the HTML to be parsed

The optional encoding parameter must be a string that indicates the encoding. If specified, that encoding will be used, regardless of any BOM or later declaration (such as in a meta element)

scripting - treat noscript elements as if javascript was turned on

parseError(*errorcode='XXX-undefined-error'*, *datavars=None*)

parseFragment(*stream*, **args*, ***kwargs*)
Parse a HTML fragment into a well-formed tree fragment

container - name of the element we're setting the innerHTML property if set to None, default to 'div'

stream - a filelike object or string containing the HTML to be parsed

The optional encoding parameter must be a string that indicates the encoding. If specified, that encoding will be used, regardless of any BOM or later declaration (such as in a meta element)

scripting - treat noscript elements as if javascript was turned on

parseRCDataRawtext(*token*, *contentType*)
Generic RCDATA/RAWTEXT Parsing algorithm contentType - RCDATA or RAWTEXT

reparseTokenNormal(*token*)

reset()

resetInsertionMode()

html5lib.__init__.**parse**(*doc*, *treebuilder='etree'*, *namespaceHTMLElements=True*, ***kwargs*)
Parse a string or file-like object into a tree

html5lib.__init__.**parseFragment**(*doc*, *container='div'*, *treebuilder='etree'*, *namespaceHTMLElements=True*, ***kwargs*)

html5lib.__init__.**getTreeBuilder**(*treeType*, *implementation=None*, ***kwargs*)
Get a TreeBuilder class for various types of tree with built-in support

treeType - the name of the tree type required (case-insensitive). Supported values are:

"dom" - A generic builder for DOM implementations, defaulting to a xml.dom.minidom based implementation.

"etree" - A generic builder for tree implementations exposing an ElementTree-like interface, defaulting to xml.etree.cElementTree if available and xml.etree.ElementTree if not.

"lxml" - A etree-based builder for lxml.etree, handling limitations of lxml's implementation.

implementation - (Currently applies to the "etree" and "dom" tree types). A module implementing the tree type e.g. xml.etree.ElementTree or xml.etree.cElementTree.

html5lib.__init__.**getTreeWalker**(*treeType*, *implementation=None*, ***kwargs*)
Get a TreeWalker class for various types of tree with built-in support

Args:

treeType (str): the name of the tree type required (case-insensitive). Supported values are:

- "dom": The xml.dom.minidom DOM implementation
- **"etree": A generic walker for tree implementations exposing an** elementtree-like interface (known to work with ElementTree, cElementTree and lxml.etree).
- "lxml": Optimized walker for lxml.etree
- "genshi": a Genshi stream

Implementation: A module implementing the tree type e.g. xml.etree.ElementTree or cElementTree (Currently applies to the "etree" tree type only).

html5lib.__init__.**serialize**(*input*, *tree='etree'*, *encoding=None*, ***serializer_opts*)

constants Module

exception html5lib.constants.**DataLossWarning**
Bases: UserWarning

exception html5lib.constants.**ReparseException**
Bases: Exception

html5parser Module

class html5lib.html5parser.**HTMLParser**(*tree=None*, *strict=False*, *namespaceHTMLElements=True*, *debug=False*)

Bases: object

HTML parser. Generates a tree structure from a stream of (possibly malformed) HTML

adjustForeignAttributes(*token*)

adjustMathMLAttributes(*token*)

adjustSVGAttributes(*token*)

documentEncoding
The name of the character encoding that was used to decode the input stream, or None if that is not determined yet.

isHTMLIntegrationPoint(*element*)

isMathMLTextIntegrationPoint(*element*)

mainLoop()

normalizeToken(*token*)
HTML5 specific normalizations to the token stream

normalizedTokens()

parse(*stream*, **args*, ***kwargs*)
Parse a HTML document into a well-formed tree

stream - a filelike object or string containing the HTML to be parsed

The optional encoding parameter must be a string that indicates the encoding. If specified, that encoding will be used, regardless of any BOM or later declaration (such as in a meta element)

scripting - treat noscript elements as if javascript was turned on

parseError(*errorcode='XXX-undefined-error'*, *datavars=None*)

parseFragment(*stream*, **args*, ***kwargs*)
Parse a HTML fragment into a well-formed tree fragment

container - name of the element we're setting the innerHTML property if set to None, default to 'div'

stream - a filelike object or string containing the HTML to be parsed

The optional encoding parameter must be a string that indicates the encoding. If specified, that encoding will be used, regardless of any BOM or later declaration (such as in a meta element)

scripting - treat noscript elements as if javascript was turned on

parseRCDataRawtext(*token*, *contentType*)
Generic RCDATA/RAWTEXT Parsing algorithm contentType - RCDATA or RAWTEXT

reparseTokenNormal(*token*)

reset()

resetInsertionMode()

exception html5lib.html5parser.**ParseError**
Bases: Exception

Error in parsed document

html5lib.html5parser.**adjust_attributes**(*token*, *replacements*)

html5lib.html5parser.**impliedTagToken**(*name*, *type='EndTag'*, *attributes=None*, *selfClosing=False*)

html5lib.html5parser.**method_decorator_metaclass**(*function*)

html5lib.html5parser.**parse**(*doc*, *treebuilder='etree'*, *namespaceHTMLElements=True*, ***kwargs*)
Parse a string or file-like object into a tree

html5lib.html5parser.**parseFragment**(*doc*, *container='div'*, *treebuilder='etree'*, *namespaceHTMLElements=True*, ***kwargs*)

serializer Module

class html5lib.serializer.**HTMLSerializer**(***kwargs*)
Bases: object

alphabetical_attributes = False

encode(*string*)

encodeStrict(*string*)

escape_lt_in_attrs = False

escape_rcdata = False

inject_meta_charset = True

minimize_boolean_attributes = True

omit_optional_tags = True

options = ('quote_attr_values', 'quote_char', 'use_best_quote_char', 'omit_optional_tags', 'minimize_boolean_attribut

quote_attr_values = 'legacy'

quote_char = '"'

render(*treewalker*, *encoding=None*)

resolve_entities = True

sanitize = False

serialize(*treewalker*, *encoding=None*)

serializeError(*data='XXX ERROR MESSAGE NEEDED'*)

space_before_trailing_solidus = True

strip_whitespace = False

use_best_quote_char = True

use_trailing_solidus = False

exception html5lib.serializer.**SerializeError**
Bases: Exception

Error in serialized tree

html5lib.serializer.**htmlentityreplace_errors**(*exc*)

html5lib.serializer.**serialize**(*input*, *tree='etree'*, *encoding=None*, ***serializer_opts*)

html5lib.serializer.**xmlcharrefreplace_errors**()
Implements the 'xmlcharrefreplace' error handling, which replaces an unencodable character with the appropriate XML character reference.

Subpackages

filters Package

base Module

class html5lib.filters.base.**Filter**(*source*)
Bases: object

alphabeticalattributes Module

class html5lib.filters.alphabeticalattributes.**Filter**(*source*)
Bases: *html5lib.filters.base.Filter*

inject_meta_charset Module

class html5lib.filters.inject_meta_charset.**Filter**(*source*, *encoding*)
Bases: *html5lib.filters.base.Filter*

lint Module

class html5lib.filters.lint.**Filter**(*source*, *require_matching_tags=True*)
Bases: *html5lib.filters.base.Filter*

optionaltags Module

class html5lib.filters.optionaltags.**Filter**(*source*)
Bases: *html5lib.filters.base.Filter*

is_optional_end(*tagname*, *next*)

is_optional_start(*tagname*, *previous*, *next*)

slider()

sanitizer Module

class html5lib.filters.sanitizer.**Filter**(*source*, *allowed_elements=frozenset({('http://www.w3.org/2000/svg', 'switch'), ('http://www.w3.org/1999/xhtml', 'hr'), ('http://www.w3.org/1999/xhtml', 'center'), ('http://www.w3.org/1999/xhtml', 'tr'), ('http://www.w3.org/1999/xhtml', 'map'), ('http://www.w3.org/1999/xhtml', 'em'), ('http://www.w3.org/1999/xhtml', 'audio'), ('http://www.w3.org/1998/Math/MathML', 'mtr'), ('http://www.w3.org/1998/Math/MathML', 'munder'), ('http://www.w3.org/1999/xhtml', 'h1'), ('http://www.w3.org/1999/xhtml', 'output'), ('http://www.w3.org/1999/xhtml', 'u'), ('http://www.w3.org/1999/xhtml', 'table'), ('http://www.w3.org/1999/xhtml', 'ol'), ('http://www.w3.org/1999/xhtml', 'h6'), ('http://www.w3.org/1998/Math/MathML', 'math'), ('http://www.w3.org/2000/svg', 'set'), ('http://www.w3.org/1999/xhtml', 'datalist'), ('http://www.w3.org/1999/xhtml', 'del'), ('http://www.w3.org/1999/xhtml', 'video'), ('http://www.w3.org/1998/Math/MathML', 'mspace'), ('http://www.w3.org/1998/Math/MathML', 'msup'), ('http://www.w3.org/2000/svg', 'radialGradient'), ('http://www.w3.org/1998/Math/MathML', 'mn'), ('http://www.w3.org/2000/svg', 'circle'), ('http://www.w3.org/1998/Math/MathML', 'mpadded'), ('http://www.w3.org/1998/Math/MathML', 'none'), ('http://www.w3.org/1999/xhtml', 'code'), ('http://www.w3.org/1999/xhtml', 'm'), ('http://www.w3.org/2000/svg', 'animate'), ('http://www.w3.org/1999/xhtml', 'details'), ('http://www.w3.org/1999/xhtml', 'dd'), ('http://www.w3.org/1999/xhtml', 'tbody'), ('http://www.w3.org/2000/svg', 'g'), ('http://www.w3.org/1999/xhtml', 'colgroup'), ('http://www.w3.org/1998/Math/MathML', 'maction'), ('http://www.w3.org/1999/xhtml', 'figcaption'), ('http://www.w3.org/1998/Math/MathML', 'mtd'), ('http://www.w3.org/1999/xhtml', 'command'), ('http://www.w3.org/1999/xhtml', 'li'), ('http://www.w3.org/2000/svg', 'polyline'), ('http://www.w3.org/1999/xhtml', 'q'), ('http://www.w3.org/1999/xhtml', 'input'), ('http://www.w3.org/1998/Math/MathML', 'mprescripts'), ('http://www.w3.org/1998/Math/MathML', 'mstyle'), ('http://www.w3.org/2000/svg', 'defs'), ('http://www.w3.org/1998/Math/MathML', 'munderover'), ('http://www.w3.org/1999/xhtml', 'button'), ('http://www.w3.org/1999/xhtml', 'br'), ('http://www.w3.org/1999/xhtml', 'select'), ('http://www.w3.org/2000/svg', 'glyph'), ('http://www.w3.org/1999/xhtml', 'blockquote'), ('http://www.w3.org/1999/xhtml', 'menu'), ('http://www.w3.org/2000/svg', 'tspan'), ('http://www.w3.org/1999/xhtml', 'spacer'), ('http://www.w3.org/1998/Math/MathML', 'mfrac'), ('http://www.w3.org/1999/xhtml', 'aside'), ('http://www.w3.org/1999/xhtml',*

sanitization of XHTML+MathML+SVG and of inline style attributes.

allowed_token(*token*)

disallowed_token(*token*)

sanitize_css(*style*)

sanitize_token(*token*)

whitespace Module

class html5lib.filters.whitespace.**Filter**(*source*)
Bases: *html5lib.filters.base.Filter*

spacePreserveElements = frozenset({'noframes', 'noembed', 'pre', 'script', 'style', 'noscript', 'iframe', 'xmp', 'texta

html5lib.filters.whitespace.**collapse_spaces**(*text*)

treebuilders Package

treebuilders Package

A collection of modules for building different kinds of tree from HTML documents.

To create a treebuilder for a new type of tree, you need to do implement several things:

1) A set of classes for various types of elements: Document, Doctype, Comment, Element. These must implement the interface of _base.treebuilders.Node (although comment nodes have a different signature for their constructor, see treebuilders.etree.Comment) Textual content may also be implemented as another node type, or not, as your tree implementation requires.

2) A treebuilder object (called TreeBuilder by convention) that inherits from treebuilders._base.TreeBuilder. This has 4 required attributes: documentClass - the class to use for the bottommost node of a document elementClass - the class to use for HTML Elements commentClass - the class to use for comments doctypeClass - the class to use for doctypes It also has one required method: getDocument - Returns the root node of the complete document tree

3) If you wish to run the unit tests, you must also create a testSerializer method on your treebuilder which accepts a node and returns a string containing Node and its children serialized according to the format used in the unittests

html5lib.treebuilders.**getTreeBuilder**(*treeType*, *implementation=None*, ***kwargs*)
Get a TreeBuilder class for various types of tree with built-in support

treeType - the name of the tree type required (case-insensitive). Supported values are:

"dom" - A generic builder for DOM implementations, defaulting to a xml.dom.minidom based implementation.

"etree" - A generic builder for tree implementations exposing an ElementTree-like interface, defaulting to xml.etree.cElementTree if available and xml.etree.ElementTree if not.

"lxml" - A etree-based builder for lxml.etree, handling limitations of lxml's implementation.

implementation - (Currently applies to the "etree" and "dom" tree types). A module implementing the tree type e.g. xml.etree.ElementTree or xml.etree.cElementTree.

base Module

class html5lib.treebuilders.base.**ActiveFormattingElements**
Bases: list

append(*node*)

nodesEqual(*node1*, *node2*)

class html5lib.treebuilders.base.**Node**(*name*)
Bases: object

appendChild(*node*)
Insert node as a child of the current node

cloneNode()
Return a shallow copy of the current node i.e. a node with the same name and attributes but with no parent or child nodes

hasContent()
Return true if the node has children or text, false otherwise

insertBefore(*node*, *refNode*)
Insert node as a child of the current node, before refNode in the list of child nodes. Raises ValueError if refNode is not a child of the current node

insertText(*data*, *insertBefore=None*)
Insert data as text in the current node, positioned before the start of node insertBefore or to the end of the node's text.

removeChild(*node*)
Remove node from the children of the current node

reparentChildren(*newParent*)
Move all the children of the current node to newParent. This is needed so that trees that don't store text as nodes move the text in the correct way

class html5lib.treebuilders.base.**TreeBuilder**(*namespaceHTMLElements*)
Bases: object

Base treebuilder implementation documentClass - the class to use for the bottommost node of a document elementClass - the class to use for HTML Elements commentClass - the class to use for comments doctypeClass - the class to use for doctypes

clearActiveFormattingElements()

commentClass = None

createElement(*token*)
Create an element but don't insert it anywhere

doctypeClass = None

documentClass = None

elementClass = None

elementInActiveFormattingElements(*name*)
Check if an element exists between the end of the active formatting elements and the last marker. If it does, return it, else return false

elementInScope(*target*, *variant=None*)

fragmentClass = None

generateImpliedEndTags(*exclude=None*)

getDocument()
Return the final tree

getFragment()
Return the final fragment

getTableMisnestedNodePosition()
Get the foster parent element, and sibling to insert before (or None) when inserting a misnested table node

insertComment(*token*, *parent=None*)

insertDoctype(*token*)

insertElementNormal(*token*)

insertElementTable(*token*)
Create an element and insert it into the tree

insertFromTable

insertRoot(*token*)

insertText(*data*, *parent=None*)
Insert text data.

reconstructActiveFormattingElements()

reset()

testSerializer(*node*)
Serialize the subtree of node in the format required by unit tests node - the node from which to start serializing

dom Module

html5lib.treebuilders.dom.**getDomBuilder**(*DomImplementation*)

etree Module

html5lib.treebuilders.etree.**getETreeBuilder**(*ElementTreeImplementation*, *fullTree=False*)

etree_lxml Module

Module for supporting the lxml.etree library. The idea here is to use as much of the native library as possible, without using fragile hacks like custom element names that break between releases. The downside of this is that we cannot represent all possible trees; specifically the following are known to cause problems:

Text or comments as siblings of the root element Docypes with no name

When any of these things occur, we emit a DataLossWarning

class html5lib.treebuilders.etree_lxml.**Document**
Bases: object

appendChild(*element*)

childNodes

class html5lib.treebuilders.etree_lxml.**DocumentType**(*name*, *publicId*, *systemId*)
Bases: object

class html5lib.treebuilders.etree_lxml.**TreeBuilder**(*namespaceHTMLElements*, *fullTree=False*)
Bases: *html5lib.treebuilders.base.TreeBuilder*

commentClass = None

doctypeClass
alias of *DocumentType*

documentClass
alias of *Document*

elementClass = None

fragmentClass
alias of *Document*

getDocument()

getFragment()

implementation = <module 'lxml.etree' from '/usr/lib/python3/dist-packages/lxml/etree.cpython-35m-x86_64-linux-gn

insertCommentInitial(*data*, *parent=None*)

insertCommentMain(*data*, *parent=None*)

insertDoctype(*token*)

insertRoot(*token*)
Create the document root

reset()

testSerializer(*element*)

html5lib.treebuilders.etree_lxml.**testSerializer**(*element*)

html5lib.treebuilders.etree_lxml.**tostring**(*element*)
Serialize an element and its child nodes to a string

treewalkers Package

treewalkers Package

A collection of modules for iterating through different kinds of tree, generating tokens identical to those produced by the tokenizer module.

To create a tree walker for a new type of tree, you need to do implement a tree walker object (called TreeWalker by convention) that implements a 'serialize' method taking a tree as sole argument and returning an iterator generating tokens.

html5lib.treewalkers.**getTreeWalker**(*treeType*, *implementation=None*, ***kwargs*)
Get a TreeWalker class for various types of tree with built-in support

Args:

treeType (str): the name of the tree type required (case-insensitive). Supported values are:

- "dom": The xml.dom.minidom DOM implementation

- **"etree": A generic walker for tree implementations exposing an** elementtree-like interface (known to work with ElementTree, cElementTree and lxml.etree).
- "lxml": Optimized walker for lxml.etree
- "genshi": a Genshi stream

Implementation: A module implementing the tree type e.g. xml.etree.ElementTree or cElementTree (Currently applies to the "etree" tree type only).

html5lib.treewalkers.**pprint**(*walker*)
Pretty printer for tree walkers

base Module

class html5lib.treewalkers.base.**TreeWalker**(*tree*)
Bases: object

comment(*data*)

doctype(*name*, *publicId=None*, *systemId=None*)

emptyTag(*namespace*, *name*, *attrs*, *hasChildren=False*)

endTag(*namespace*, *name*)

entity(*name*)

error(*msg*)

startTag(*namespace*, *name*, *attrs*)

text(*data*)

unknown(*nodeType*)

class html5lib.treewalkers.base.**NonRecursiveTreeWalker**(*tree*)
Bases: *html5lib.treewalkers.base.TreeWalker*

getFirstChild(*node*)

getNextSibling(*node*)

getNodeDetails(*node*)

getParentNode(*node*)

dom Module

class html5lib.treewalkers.dom.**TreeWalker**(*tree*)
Bases: *html5lib.treewalkers.base.NonRecursiveTreeWalker*

getFirstChild(*node*)

getNextSibling(*node*)

getNodeDetails(*node*)

getParentNode(*node*)

etree Module

html5lib.treewalkers.etree.**getETreeBuilder**(*ElementTreeImplementation*)

etree_lxml Module

class html5lib.treewalkers.etree_lxml.**Doctype**(*root_node*, *name*, *public_id*, *system_id*)
Bases: object

getnext()

class html5lib.treewalkers.etree_lxml.**FragmentRoot**(*children*)
Bases: *html5lib.treewalkers.etree_lxml.Root*

getnext()

class html5lib.treewalkers.etree_lxml.**FragmentWrapper**(*fragment_root*, *obj*)
Bases: object

getnext()

getparent()

class html5lib.treewalkers.etree_lxml.**Root**(*et*)
Bases: object

getnext()

class html5lib.treewalkers.etree_lxml.**TreeWalker**(*tree*)
Bases: *html5lib.treewalkers.base.NonRecursiveTreeWalker*

getFirstChild(*node*)

getNextSibling(*node*)

getNodeDetails(*node*)

getParentNode(*node*)

html5lib.treewalkers.etree_lxml.**ensure_str**(*s*)

genshi Module

class html5lib.treewalkers.genshi.**TreeWalker**(*tree*)
Bases: *html5lib.treewalkers.base.TreeWalker*

tokens(*event*, *next*)

Change Log

0.999999999/1.0b10

Released on July 15, 2016

- Fix attribute order going to the tree builder to be document order instead of reverse document order(!).

0.99999999/1.0b9

Released on July 14, 2016

- **Added ordereddict as a mandatory dependency on Python 2.6.**
- Added `lxml`, `genshi`, `datrie`, `charade`, and `all` extras that will do the right thing based on the specific interpreter implementation.
- Now requires the `mock` package for the testsuite.
- Cease supporting DATrie under PyPy.
- **Remove ‘‘PullDOM‘‘ support, as this hasn’t ever been properly tested, doesn’t entirely work, and as far as I can tell is completely unused by anyone.**
- Move testsuite to `py.test`.
- **Fix #124: move to webencodings for decoding the input byte stream; this makes html5lib compliant with the Encoding Standard, and introduces a required dependency on webencodings.**
- **Cease supporting Python 3.2 (in both CPython and PyPy forms).**
- **Fix comments containing double-dash with lxml 3.5 and above.**
- **Use scripting disabled by default (as we don’t implement scripting).**
- **Fix #11, avoiding the XSS bug potentially caused by serializer allowing attribute values to be escaped out of in old browser versions, changing the quote_attr_values option on serializer to take one of three values, “always” (the old True value), “legacy” (the new option, and the new default), and “spec” (the old False value, and the old default).**
- **Fix #72 by rewriting the sanitizer to apply only to treewalkers (instead of the tokenizer); as such, this will require amending all callers of it to use it via the treewalker API.**
- **Drop support of charade, now that chardet is supported once more.**
- **Replace the charset keyword argument on parse and related methods with a set of keyword arguments: override_encoding, transport_encoding, same_origin_parent_encoding, likely_encoding, and default_encoding.**
- **Move filters._base, treebuilder._base, and treewalkers._base to .base to clarify their status as public.**
- **Get rid of the sanitizer package. Merge sanitizer.sanitize into the sanitizer.htmlsanitizer module and move that to saniziter. This means anyone who used sanitizer.sanitize or sanitizer.HTMLSanitizer needs no code changes.**
- **Rename treewalkers.lxmletree to .etree_lxml and treewalkers.genshistream to .genshi to have a consistent API.**
- Move a whole load of stuff (inputstream, ihatexml, trie, tokenizer, utils) to be underscore prefixed to clarify their status as private.

0.9999999/1.0b8

Released on September 10, 2015

- Fix #195: fix the sanitizer to drop broken URLs (it threw an exception between 0.9999 and 0.999999).

0.999999/1.0b7

Released on July 7, 2015

- Fix #189: fix the sanitizer to allow relative URLs again (as it did prior to 0.9999/1.0b5).

0.99999/1.0b6

Released on April 30, 2015

- Fix #188: fix the sanitizer to not throw an exception when sanitizing bogus data URLs.

0.9999/1.0b5

Released on April 29, 2015

- Fix #153: Sanitizer fails to treat some attributes as URLs. Despite how this sounds, this has no known security implications. No known version of IE (5.5 to current), Firefox (3 to current), Safari (6 to current), Chrome (1 to current), or Opera (12 to current) will run any script provided in these attributes.
- Pass error message to the ParseError exception in strict parsing mode.
- Allow data URIs in the sanitizer, with a whitelist of content-types.
- Add support for Python implementations that don't support lone surrogates (read: Jython). Fixes #2.
- Remove localization of error messages. This functionality was totally unused (and untested that everything was localizable), so we may as well follow numerous browsers in not supporting translating technical strings.
- Expose treewalkers.pprint as a public API.
- Add a documentEncoding property to HTML5Parser, fix #121.

0.999

Released on December 23, 2013

- Fix #127: add work-around for CPython issue #20007: .read(0) on http.client.HTTPResponse drops the rest of the content.
- Fix #115: lxml treewalker can now deal with fragments containing, at their root level, text nodes with non-ASCII characters on Python 2.

0.99

Released on September 10, 2013

- No library changes from 1.0b3; released as 0.99 as pip has changed behaviour from 1.4 to avoid installing pre-release versions per PEP 440.

1.0b3

Released on July 24, 2013

- Removed `RecursiveTreeWalker` from `treewalkers._base`. Any implementation using it should be moved to `NonRecursiveTreeWalker`, as everything bundled with html5lib has for years.

- Fix #67 so that `BufferedStream` to correctly returns a bytes object, thereby fixing any case where html5lib is passed a non-seekable RawIOBase-like object.

1.0b2

Released on June 27, 2013

- Removed reordering of attributes within the serializer. There is now an `alphabetical_attributes` option which preserves the previous behaviour through a new filter. This allows attribute order to be preserved through html5lib if the tree builder preserves order.
- Removed `dom2sax` from DOM treebuilders. It has been replaced by `treeadapters.sax.to_sax` which is generic and supports any treewalker; it also resolves all known bugs with `dom2sax`.
- Fix treewalker assertions on hitting bytes strings on Python 2. Previous to 1.0b1, treewalkers coped with mixed bytes/unicode data on Python 2; this reintroduces this prior behaviour on Python 2. Behaviour is unchanged on Python 3.

1.0b1

Released on May 17, 2013

- Implementation updated to implement the HTML specification as of 5th May 2013 (SVN revision r7867).
- Python 3.2+ supported in a single codebase using the `six` library.
- Removed support for Python 2.5 and older.
- Removed the deprecated Beautiful Soup 3 treebuilder. `beautifulsoup4` can use `html5lib` as a parser instead. Note that since it doesn't support namespaces, foreign content like SVG and MathML is parsed incorrectly.
- Removed `simpletree` from the package. The default tree builder is now `etree` (using the `xml.etree.cElementTree` implementation if available, and `xml.etree.ElementTree` otherwise).
- Removed the `XHTMLSerializer` as it never actually guaranteed its output was well-formed XML, and hence provided little of use.
- Removed default DOM treebuilder, so `html5lib.treebuilders.dom` is no longer supported. `html5lib.treebuilders.getTreeBuilder("dom")` will return the default DOM treebuilder, which uses `xml.dom.minidom`.
- Optional heuristic character encoding detection now based on `charade` for Python 2.6 - 3.3 compatibility.
- Optional `Genshi` treewalker support fixed.
- Many bugfixes, including:
  - #33: null in attribute value breaks XML AttValue;
  - #4: nested, indirect descendant, <button> causes infinite loop;
  - Google Code 215: Properly detect seekable streams;
  - Google Code 206: add support for <video preload=...>, <audio preload=...>;
  - Google Code 205: add support for <video poster=...>;
  - Google Code 202: Unicode file breaks InputStream.
- Source code is now mostly PEP 8 compliant.
- Test harness has been improved and now depends on `nose`.

- Documentation updated and moved to https://html5lib.readthedocs.io/.

0.95

Released on February 11, 2012

0.90

Released on January 17, 2010

0.11.1

Released on June 12, 2008

0.11

Released on June 10, 2008

0.10

Released on October 7, 2007

0.9

Released on March 11, 2007

0.2

Released on January 8, 2007

License

CHAPTER 7

Indices and tables

- genindex
- modindex
- search

Python Module Index

h

Index

F

G

H

I

M

N

O

P

Q

R

S

T

U

X