htmlmin Documentation

Release 0.1

Dave Mankoff

Dec 07, 2017

Contents

An HTML Minifier with Seatbelts

CHAPTER 1

Quickstart

For single invocations, there is the *htmlmin.minify* method. It takes input html as a string for its first argument and returns minified html. It accepts multiple different options that allow you to tune the amount of minification being done, with the defaults being the safest available options:

```
>>> import htmlmin
>>> input_html = '''
  <body   style="background-color: tomato;">
    <h1>  htmlmin   rocks</h1>
    <pre>
      and rolls
    </pre>
  </body>'''
>>> htmlmin.minify(input_html)
u' <body style="background-color: tomato;"> <h1> htmlmin rocks</h1> <pre>\n      ␣
↪and rolls\n      </pre> </body>'
>>> print htmlmin.minify(input_html)
 <body style="background-color: tomato;"> <h1> htmlmin rocks</h1> <pre>
        and rolls
      </pre> </body>
```

If there is a chunk of html which you do not want minified, put a `pre` attribute on an HTML tag that wraps it. htmlmin will leave the contents of the tag alone and will remove the `pre` attribute before it is output:

```
>>> import htmlmin
>>> input_html = '''<span>   minified   </span><span pre>   not minified   </span>'''
>>> htmlmin.minify(input_html)
u'<span> minified </span><span>   not minified   </span>'
```

Attributes will be condensed to their smallest possible representation by default. You can prefix an individual attribute with `pre-` to leave it unchanged:

```
>>> import htmlmin
>>> input_html = '''<input value="&lt;minified&gt;" /><input pre-value="&lt;not␣
↪minified&gt;" />'''
```

```
>>> htmlmin.minify(input_html)
u'<input value="<minified>"><input value=&lt;not minified&gt;>'
```

The *minify* function works well for one off minifications. However, if you are going to minify several pieces of HTML, the *Minifier* class is provided. It works similarly, but allows for persistence of options between invocations and recycles the internal data structures used for minification.

1.1 Command Line

htmlmin is invoked by running:

```
htmlmin input.html output.html
```

If no output file is specified, it will print to `stdout`. If no input specified, it reads form `stdin`. Help with options can be retrieved at any time by running *htmlmin -h*:

```
htmlmin -h
usage: htmlmin [-h] [-c] [-s] [--remove-all-empty-space]
               [--keep-optional-attribute-quotes] [-H] [-k] [-a PRE_ATTR]
               [-p [TAG [TAG ...]]] [-e ENCODING]
               [INPUT] [OUTPUT]

Minify HTML

positional arguments:
  INPUT                 File path to html file to minify. Defaults to stdin.
  OUTPUT                File path to output to. Defaults to stdout.

optional arguments:
  -h, --help            show this help message and exit
  -c, --remove-comments
                        When set, comments will be removed. They can be kept on an␣
↪individual basis
                        by starting them with a '!': <!--! comment -->. The '!' will␣
↪be removed from
                        the final output. If you want a '!' as the leading character␣
↪of your comment,
                        put two of them: <!--!! comment -->.

  -s, --remove-empty-space
                        When set, this removes empty space betwen tags in certain␣
↪cases.
                        Specifically, it will remove empty space if and only if there␣
↪a newline
                        character occurs within the space. Thus, code like
                        '<span>x</span> <span>y</span>' will be left alone, but code␣
↪such as
                        '   ...
                          </head>
                          <body>
                            ...'
                        will become '...</head><body>...'. Note that this CAN break␣
↪your
                        html if you spread two inline tags over two lines. Use with␣
↪caution.
```

```
  --remove-all-empty-space
                        When set, this removes ALL empty space betwen tags. WARNING:␣
↪this can and
                        likely will cause unintended consequences. For instance, '<i>X
↪</i> <i>Y</i>'
                        will become '<i>X</i><i>Y</i>'. Putting whitespace along with␣
↪other text will
                        avoid this problem. Only use if you are confident in the␣
↪result. Whitespace is
                        not removed from inside of tags, thus '<span> </span>' will␣
↪be left alone.

  --keep-optional-attribute-quotes
                        When set, this keeps all attribute quotes, even if they are␣
↪optional.

  -H, --in-head         If you are parsing only a fragment of HTML, and the fragment␣
↪occurs in the
                        head of the document, setting this will remove some extra␣
↪whitespace.

  -k, --keep-pre-attr   HTMLMin supports the propietary attribute 'pre' that can be␣
↪added to elements
                        to prevent minification. This attribute is removed by default.
↪ Set this flag to
                        keep the 'pre' attributes in place.

  -a PRE_ATTR, --pre-attr PRE_ATTR
                        The attribute htmlmin looks for to find blocks of HTML that␣
↪it should not
                        minify. This attribute will be removed from the HTML unless '-
↪k' is
                        specified. Defaults to 'pre'. You can also prefix individual␣
↪tag attributes
                        with ``{pre_attr}-`` to prevent the contents of the␣
↪individual attribute from
                        being changed.

  -p [TAG [TAG ...]], --pre-tags [TAG [TAG ...]]
                        By default, the contents of 'pre', and 'textarea' tags are␣
↪left unminified.
                        You can specify different tags using the --pre-tags option.
↪'script' and 'style'
                        tags are always left unmininfied.

  -e ENCODING, --encoding ENCODING
                        Encoding to read and write with. Default 'utf-8'.
```

CHAPTER 2

Tutorial & Examples

Coming soon...

CHAPTER 3

API Reference

3.1 Main Functions

`htmlmin.minify`(*input*, *remove_comments=False*, *remove_empty_space=False*, *remove_all_empty_space=False*, *reduce_empty_attributes=True*, *reduce_boolean_attributes=False*, *remove_optional_attribute_quotes=True*, *convert_charrefs=True*, *keep_pre=False*, *pre_tags=(u'pre'*, *u'textarea')*, *pre_attr='pre'*, *cls=<class htmlmin.parser.HTMLMinParser>*)

Minifies HTML in one shot.

Parameters

- **input** – A string containing the HTML to be minified.
- **remove_comments** – Remove comments found in HTML. Individual comments can be maintained by putting a `!` as the first character inside the comment. Thus:

  ```
  <!-- FOO --> <!--! BAR -->
  ```

  Will become simply:

  ```
  <!-- BAR -->
  ```

  The added exclamation is removed.
- **remove_empty_space** – Remove empty space found in HTML between an opening and a closing tag and when it contains a newline or carriage return. If whitespace is found that is only spaces and/or tabs, it will be turned into a single space. Be careful, this can have unintended consequences.
- **remove_all_empty_space** – A more extreme version of `remove_empty_space`, this removes all empty whitespace found between tags. This is almost guaranteed to break your HTML unless you are very careful.
- **reduce_boolean_attributes** – Where allowed by the HTML5 specification, attributes such as 'disabled' and 'readonly' will have their value removed, so 'dis-

abled="true"' will simply become 'disabled'. This is generally a good option to turn on except when JavaScript relies on the values.

- **`remove_optional_attribute_quotes`** – When True, optional quotes around attributes are removed. When False, all attribute quotes are left intact. Defaults to True.
- **`conver_charrefs`** – Decode character references such as &amp; and &#46; to their single charater values where safe. This currently only applies to attributes. Data content between tags will be left encoded.
- **`keep_pre`** – By default, htmlmin uses the special attribute `pre` to allow you to demarcate areas of HTML that should not be minified. It removes this attribute as it finds it. Setting this value to `True` tells htmlmin to leave the attribute in the output.
- **`pre_tags`** – A list of tag names that should never be minified. You are free to change this list as you see fit, but you will probably want to include `pre` and `textarea` if you make any changes to the list. Note that `<script>` and `<style>` tags are never minimized.
- **`pre_attr`** – Specifies the attribute that, when found in an HTML tag, indicates that the content of the tag should not be minified. Defaults to `pre`. You can also prefix individual tag attributes with `{pre_attr}-` to prevent the contents of the individual attribute from being changed.

Returns A string containing the minified HTML.

If you are going to be minifying multiple HTML documents, each with the same settings, consider using *`Minifier`*.

class `htmlmin.`**`Minifier`**(*remove_comments=False*, *remove_empty_space=False*, *remove_all_empty_space=False*, *reduce_empty_attributes=True*, *reduce_boolean_attributes=False*, *remove_optional_attribute_quotes=True*, *convert_charrefs=True*, *keep_pre=False*, *pre_tags=(u'pre', u'textarea')*, *pre_attr='pre'*, *cls=<class htmlmin.parser.HTMLMinParser>*)

An object that supports HTML Minification.

Options are passed into this class at initialization time and are then persisted across each use of the instance. If you are going to be minifying multiple peices of HTML, this will be more efficient than using *`htmlmin.minify`*.

See *`htmlmin.minify`* for an explanation of options.

`minify`(**input*)

Runs HTML through the minifier in one pass.

Parameters **`input`** – HTML to be fed into the minimizer. Multiple chunks of HTML can be provided, and they are fed in sequentially as if they were concatenated.

Returns A string containing the minified HTML.

This is the simplest way to use an existing `Minifier` instance. This method takes in HTML and minfies it, returning the result. Note that this method resets the internal state of the parser before it does any work. If there is pending HTML in the buffers, it will be lost.

`input`(**input*)

Feed more HTML into the input stream

Parameters **`input`** – HTML to be fed into the minimizer. Multiple chunks of HTML can be provided, and they are fed in sequentially as if they were concatenated. You can also call this method multiple times to achieve the same effect.

`output`

Retrieve the minified output generated thus far.

finalize()

Finishes current input HTML and returns mininified result.

This method flushes any remaining input HTML and returns the minified result. It resets the state of the internal parser in the process so that new HTML can be minified. Be sure to call this method before you reuse the `Minifier` instance on a new HTML document.

3.2 WSGI Middlware

class `htmlmin.middleware.`**HTMLMinMiddleware**(*app*, *by_default=True*, *keep_header=False*, *debug=False*, ***kwargs*)

WSGI Middleware that minifies html on the way out.

Parameters

- **by_default** – Specifies if minification should be turned on or off by default. Defaults to `True`.
- **keep_header** – The middleware recognizes one custom HTTP header that can be used to turn minification on or off on a per-request basis: `X-HTML-Min-Enable`. Setting the header to `true` will turn minfication on; anything else will turn minification off. If `by_default` is set to `False`, this header is how you would turn minification back on. The middleware, by default, removes the header from the output. Setting this to `True` leaves the header in tact.
- **debug** – A quick setting to turn all minification off. The middleware is effectively bypassed.

This simple middleware minifies any HTML content that passes through it. Any additional keyword arguments beyond the three settings the middleware has are passed on to the internal minifier. The documentation for the options can be found under *htmlmin.minify*.

3.3 Decorator

`htmlmin.decorator.`**htmlmin**(**args*, ***kwargs*)

Minifies HTML that is returned by a function.

A simple decorator that minifies the HTML output of any function that it decorates. It supports all the same options that *htmlmin.minify* has. With no options, it uses `minify`'s default settings:

```
@htmlmin
def foobar():
   return '   minify me!   '
```

or:

```
@htmlmin(remove_comments=True)
def foobar():
   return '   minify me!  <!-- and remove me! -->'
```

htmlmin is an HTML minifier that just works. It comes with safe defaults and an easily configurable set options. It can turn this:

```
<html>
  <head>
```

```
    <title>  Hello, World!  </title>
  </head>
  <body>
    <p> How are <em>you</em> doing?  </p>
  </body>
</html>
```

Into this:

```
<html><head><title>Hello, World!</title><body><p> How are <em>you</em> doing? </p></
↪body></html>
```

When we say that htmlmin has 'seatbelts', what we mean is that it comes with features that you can use to safely minify beyond the defaults, but you have to put them in yourself. For instance, by default, htmlmin will never minimize the content between `<pre>`, `<textarea>`, `<script>`, and `<style>` tags. You can also explicitly tell it to not minify additional tags either globally by name or by adding the custom `pre` attribute to a tag in your HTML. htmlmin will remove the `pre` attributes as it parses your HTML automatically.

It also includes a command-line tool for easy invocation and integration with existing workflows.

CHAPTER 4

Install

To install via pip:

```
pip install htmlmin
```

CHAPTER 5

Source Code

Source code is availble on github at https://github.com/mankyd/htmlmin:

```
git clone git://github.com/mankyd/htmlmin.git
```

CHAPTER 6

Features

- Safely minify HTML with either a `function call` or from the *command line*.
- Extend what elements can and cannot be minified.
- Intelligently remove whitespace completely or reduce to single spaces.
- Properly handles unclosed HTML5 tags.
- Optionally remove comments while marking some comments to keep.
- Simple function `decorator` to minify all function output.
- Simple `WSGI middleware` to minify web app output.
- Tested in both Python 2.7 and 3.2:

CHAPTER 7

Indices and tables

- genindex
- search

Index