
cssselect2

Release 0.2.1

Oct 02, 2017

Contents

1	Installation	3
2	Basic Example	5
3	cssselect2 changelog	9
3.1	Version 0.2.1	9
3.2	Version 0.2.0	9
3.3	Version 0.1	9
	Python Module Index	11

cssselect2 is a straightforward implementation of [CSS3 Selectors](#) for markup documents (HTML, XML, etc.) that can be read by [ElementTree](#)-like parsers (including [cElementTree](#), [lxml](#), [html5lib](#), etc.)

Unlike [cssselect](#), it does not translate selectors to [XPath](#) and therefore does not have all the correctness corner cases that are hard or impossible to fix in [cssselect](#).

Quick facts:

- Free software: BSD licensed
- Compatible with Python 2.7+ and 3.3+
- Latest documentation: <http://cssselect2.readthedocs.io/>
- Source, issues and pull requests [on Github](#)
- Releases [on PyPI](#)
- Install with `pip install cssselect2`

CHAPTER 1

Installation

Installing `cssselect2` with `pip` should Just Work:

```
pip install cssselect2
```

This will also automatically install `cssselect2`'s only dependency, `tinycss2`. `cssselect2` and `tinycss2` both only contain Python code and should work on any Python implementation, although they're only tested on CPython.

Here is a classical cssselect2 workflow:

- parse a CSS stylesheet using `tinycss2`,
- store the CSS rules in a `Matcher()` object,
- parse an HTML document using an ElementTree-like parser,
- wrap the HTML tree in a `ElementWrapper()` object,
- find the CSS rules matching each HTML tag, using the matcher and the wrapper.

```
from xml.etree import ElementTree

import cssselect2
import tinycss2

# Parse CSS and add rules to the matcher
matcher = cssselect2.Matcher()

rules = tinycss2.parse_stylesheet('''
  body { font-size: 2em }
  body p { background: red }
  p { color: blue }
''', skip_whitespace=True)

for rule in rules:
    selectors = cssselect2.compile_selector_list(rule.prelude)
    selector_string = tinycss2.serialize(rule.prelude)
    content_string = tinycss2.serialize(rule.content)
    payload = (selector_string, content_string)
    for selector in selectors:
        matcher.add_selector(selector, payload)
```

```
# Parse HTML and find CSS rules applying to each tag

html_tree = ElementTree.fromstring('''
<html>
  <body>
    <p>Test</p>
  </body>
</html>
''')
wrapper = cssselect2.ElementWrapper.from_html_root(html_tree)
for element in wrapper.iter_subtree():
    tag = element.etree_element.tag.split(' ')[-1]
    print('Found tag "{}" in HTML'.format(tag))

    matches = matcher.match(element)
    if matches:
        for match in matches:
            specificity, order, pseudo, payload = match
            selector_string, content_string = payload
            print('Matching selector "{}" ({}).format(
                selector_string, content_string)
    else:
        print('No rule matching this tag')
print()
```

class `cssselect2.Matcher`

A CSS selectors storage that can match against HTML elements.

add_selector (*selector, payload*)

Add a selector and its payload to the matcher.

Parameters

- **selector** – A `CompiledSelector` object.
- **payload** – Some data associated to the selector, such as declarations parsed from the content of a style rule. It can be any Python object, and will be returned as-is by `match()`.

match (*element*)

Match selectors against the given element.

Parameters **element** – An `ElementWrapper`.

Returns A list of the `payload` objects associated to selectors that match `element`, in order of lowest to highest `specificity` and in order of addition with `add_selector()` among selectors of equal `specificity`.

`cssselect2.compile_selector_list` (*input, namespaces=None*)

Compile a (comma-separated) list of selectors.

Parameters

- **input** – A `tinycss2:string`, or an iterable of `tinycss2 tinycss2:component` values such as the prelude of a style rule.
- **namespaces** – A optional dictionary of all `namespace prefix declarations` in scope for this selector. Keys are namespace prefixes as strings, or `None` for the default namespace. Values are namespace URLs as strings. If omitted, assume that no prefix is declared.

Returns A list of opaque `CompiledSelector` objects.

class `cssselect2.ElementWrapper` (*etree_element*, *parent*, *index*, *previous*, *in_html_document*, *content_language=None*)

A wrapper for an ElementTree `Element` for Selector matching.

This class should not be instantiated directly. `from_xml_root()` or `from_html_root()` should be used for the root element of a document, and other elements should be accessed (and wrappers generated) using methods such as `iter_children()` and `iter_subtree()`.

`ElementWrapper` objects compare equal if their underlying `Element` do.

classes

The classes of this element, as a `set` of strings.

etree_children

This element's children, as a list of ElementTree `Element`.

Other ElementTree nodes such as `comments` and `processing instructions` are not included.

etree_element = None

The underlying ElementTree `Element`

etree_siblings = None

The *parent*'s children as a list of ElementTree `:class:'~xml.etree.ElementTree.Element`'s. For the root (which has no parent)

classmethod `from_html_root` (*root*, *content_language=None*)

Same as `from_xml_root()`, but for documents parsed with an HTML parser like `html5lib`, which should be the case of documents with the `text/html` MIME type.

Compared to `from_xml_root()`, this makes element attribute names in Selectors case-insensitive.

classmethod `from_xml_root` (*root*, *content_language=None*)

Wrap for selector matching the root of an XML or XHTML document.

Parameters *root* – An ElementTree `Element` for the root element of a document. If the given element is not the root, selector matching will behave as if it were. In other words, selectors will be `scope-contained` to the subtree rooted at that element.

Returns A new `ElementWrapper`

id

The ID of this element, as a string.

index = None

The position within the *parent*'s children, counting from 0. `e.etree_siblings[e.index]` is always `e.etree_element`.

iter_ancestors()

Return an iterator of existing `ElementWrapper` objects for this element's ancestors, in reversed tree order (from *parent* to the root)

The element itself is not included, this is an empty sequence for the root element.

iter_children()

Return an iterator of newly-created `ElementWrapper` objects for this element's child elements, in tree order.

iter_previous_siblings()

Return an iterator of existing `ElementWrapper` objects for this element's previous siblings, in reversed tree order.

The element itself is not included, this is an empty sequence for a first child or the root element.

iter_subtree()

Return an iterator of newly-created *ElementWrapper* objects for the entire subtree rooted at this element, in tree order.

Unlike in other methods, the element itself *is* included.

This loops over an entire document:

```
for element in ElementWrapper.from_root(root_etree).iter_subtree():
    ...
```

lang

The language of this element, as a string.

local_name

The local name of this element, as a string.

matches(*selectors)

Return whether this element matches any of the given selectors.

Parameters *selectors* – Each given selector is either a *CompiledSelector*, or an argument to *compile_selector_list()*.

namespace_url

The namespace URL of this element, as a string.

parent = None

The parent *ElementWrapper*, or *None* for the root element.

previous = None

The previous sibling *ElementWrapper*, or *None* for the root element.

query(*selectors)

Return the first element (in tree order) that matches any of the given selectors.

Parameters *selectors* – Each given selector is either a *CompiledSelector*, or an argument to *compile_selector_list()*.

Returns A newly-created *ElementWrapper* object, or *None* if there is no match.

query_all(*selectors)

Return elements, in tree order, that match any of the given selectors.

Selectors are *scope-filtered* to the subtree rooted at this element.

Parameters *selectors* – Each given selector is either a *CompiledSelector*, or an argument to *compile_selector_list()*.

Returns An iterator of newly-created *ElementWrapper* objects.

class `cssselect2.SelectorError`

A specialized *ValueError* for invalid selectors.

Version 0.2.1

Released on 2017-10-02.

- Fix documentation.

Version 0.2.0

Released on 2017-08-16.

- Fix some selectors for HTML documents with no namespace.
- Don't crash when the attribute comparator is unknown.
- Don't crash when there are empty attribute classes.
- Follow semantic versioning.

Version 0.1

Released on 2017-07-07.

- Initial release.

C

`cssselect2`, 6

A

add_selector() (cssselect2.Matcher method), 6

C

classes (cssselect2.ElementWrapper attribute), 7
compile_selector_list() (in module cssselect2), 6
cssselect2 (module), 6

E

ElementWrapper (class in cssselect2), 6
etree_children (cssselect2.ElementWrapper attribute), 7
etree_element (cssselect2.ElementWrapper attribute), 7
etree_siblings (cssselect2.ElementWrapper attribute), 7

F

from_html_root() (cssselect2.ElementWrapper class
method), 7
from_xml_root() (cssselect2.ElementWrapper class
method), 7

I

id (cssselect2.ElementWrapper attribute), 7
index (cssselect2.ElementWrapper attribute), 7
iter_ancestors() (cssselect2.ElementWrapper method), 7
iter_children() (cssselect2.ElementWrapper method), 7
iter_previous_siblings() (cssselect2.ElementWrapper
method), 7
iter_subtree() (cssselect2.ElementWrapper method), 7

L

lang (cssselect2.ElementWrapper attribute), 8
local_name (cssselect2.ElementWrapper attribute), 8

M

match() (cssselect2.Matcher method), 6
Matcher (class in cssselect2), 6
matches() (cssselect2.ElementWrapper method), 8

N

namespace_url (cssselect2.ElementWrapper attribute), 8

P

parent (cssselect2.ElementWrapper attribute), 8
previous (cssselect2.ElementWrapper attribute), 8

Q

query() (cssselect2.ElementWrapper method), 8
query_all() (cssselect2.ElementWrapper method), 8

S

SelectorError (class in cssselect2), 8