tinycss Documentation

Release 0.4

Simon Sapin

Mar 25, 2017

Contents

tinycss is a complete yet simple CSS parser for Python. It supports the full syntax and error handling for CSS 2.1 as well as some CSS 3 modules:

- CSS Color 3
- CSS Fonts 3
- CSS Paged Media 3

It is designed to be easy to extend for new CSS modules and syntax, and integrates well with cssselect for Selectors 3 support.

Quick facts:

- Free software: BSD licensed
- Compatible with Python 2.7 and 3.x
- Latest documentation on python.org
- Source, issues and pull requests on Github
- Releases on PyPI
- Install with `pip install tinycss`

CHAPTER 1

Requirements

tinycss is tested on CPython 2.7, 3.3, 3.4 and 3.5 as well as PyPy 5.3 and PyPy3 2.4; it should work on any implementation of **Python 2.7 or later version (including 3.x)** of the language.

Cython is used for optional accelerators but is only required for development versions on tinycss.

CHAPTER 2

Installation

Installing with pip should Just Work:

```
pip install tinycss
```

The release tarballs contain pre-*cythoned* C files for the accelerators: you will not need Cython to install like this. If the accelerators fail to build for some reason, tinycss will print a warning and fall back to a pure-Python installation.

CHAPTER 3

Documentation

Parsing with tinycss

Quickstart

Import *tinycss*, make a parser object with the features you want, and parse a stylesheet:

```
>>> import tinycss
>>> parser = tinycss.make_parser('page3')
>>> stylesheet = parser.parse_stylesheet_bytes(b'''@import "foo.css";
...     p.error { color: red }  @lorem-ipsum;
...     @page tables { size: landscape }''')
>>> stylesheet.rules
[<ImportRule 1:1 foo.css>, <RuleSet at 2:5 p.error>, <PageRule 3:5 ('tables', None)>]
>>> stylesheet.errors
[ParseError('Parse error at 2:29, unknown at-rule in stylesheet context: @lorem-ipsum
↪',)]
```

You'll get a `Stylesheet` object which contains all the parsed content as well as a list of encountered errors.

Parsers

Parsers are subclasses of `tinycss.css21.CSS21Parser`. Various subclasses add support for more syntax. You can choose which features to enable by making a new parser class with multiple inheritance, but there is also a convenience function to do that:

Parsing a stylesheet

Parser classes have three different methods to parse CSS stylesheet, depending on whether you have a file, a byte string, or an Unicode string.

Parsing a `style` attribute

Parsed objects

These data structures make up the results of the various parsing methods.

Note: All subsequent objects have `line` and `column` attributes (not repeated every time fore brevity) that indicate where in the CSS source this object was read.

Tokens

Some parts of a stylesheet (such as selectors in CSS 2.1 or property values) are not parsed by tinycss. They appear as tokens instead.

CSS 3 Modules

Selectors 3

On `RuleSet.selector`, the `as_css()` method can be used to serialize a selector back to an Unicode string.

```
>>> import tinycss
>>> stylesheet = tinycss.make_parser().parse_stylesheet(
...     'div.error, #root > section:first-letter { color: red }')
>>> selector_string = stylesheet.rules[0].selector.as_css()
>>> selector_string
'div.error, #root > section:first-letter'
```

This string can be parsed by cssselect. The parsed objects have information about pseudo-elements and selector specificity.

```
>>> import cssselect
>>> selectors = cssselect.parse(selector_string)
>>> [s.specificity() for s in selectors]
[(0, 1, 1), (1, 0, 2)]
>>> [s.pseudo_element for s in selectors]
[None, 'first-letter']
```

These objects can in turn be translated to XPath expressions. Note that the translation ignores pseudo-elements, you have to account for them somehow or reject selectors with pseudo-elements.

```
>>> xpath = cssselect.HTMLTranslator().selector_to_xpath(selectors[1])
>>> xpath
"descendant-or-self::*[@id = 'root']/section"
```

Finally, the XPath expressions can be used with lxml to find the matching elements.

```
>>> from lxml import etree
>>> compiled_selector = etree.XPath(xpath)
>>> document = etree.fromstring('''<section id="root">
...   <section id="head">Title</section>
...   <section id="content">
...     Lorem <section id="sub-section">ipsum</section>
```

```
...   </section>
... </section>''')
>>> [el.get('id') for el in compiled_selector(document)]
['head', 'content']
```

Find more details in the cssselect documentation.

Color 3

This module implements parsing for the <*color*> values, as defined in CSS 3 Color.

The (deprecated) CSS2 system colors are not supported, but you can easily test for them if you want as they are simple `IDENT` tokens. For example:

```
if token.type == 'IDENT' and token.value == 'ButtonText':
    return ...
```

All other values types *are* supported:

- Basic, extended (X11) and transparent color keywords;
- 3-digit and 6-digit hexadecimal notations;
- `rgb()`, `rgba()`, `hsl()` and `hsla()` functional notations.
- `currentColor`

This module does not integrate with a parser class. Instead, it provides a function that can parse tokens as found in `css21.Declaration.value`, for example.

Paged Media 3

Fonts 3

Other CSS modules

To add support for new CSS syntax, see *Extending the parser*.

Extending the parser

Modules such as *page3* extend the CSS 2.1 parser to add support for CSS 3 syntax. They do so by sub-classing `css21.CSS21Parser` and overriding/extending some of its methods. If fact, the parser is made of methods in a class (rather than a set of functions) solely to enable this kind of sub-classing.

tinycss is designed to enable you to have parser subclasses outside of tinycss, without monkey-patching. If however the syntax you added is for a W3C specification, consider including your subclass in a new tinycss module and send a pull request: see *Hacking tinycss*.

Example: star hack

The star hack uses invalid declarations that are only parsed by some versions of Internet Explorer. By default, tinycss ignores invalid declarations and logs an error.

```
>>> from tinycss.css21 import CSS21Parser
>>> css = '#elem { width: [W3C Model Width]; *width: [BorderBox Model]; }'
>>> stylesheet = CSS21Parser().parse_stylesheet(css)
>>> stylesheet.errors
[ParseError('Parse error at 1:35, expected a property name, got DELIM',)]
>>> [decl.name for decl in stylesheet.rules[0].declarations]
['width']
```

If for example a minifier based on tinycss wants to support the star hack, it can by extending the parser:

```
>>> class CSSStarHackParser(CSS21Parser):
...     def parse_declaration(self, tokens):
...         has_star_hack = (tokens[0].type == 'DELIM' and tokens[0].value == '*')
...         if has_star_hack:
...             tokens = tokens[1:]
...         declaration = super(CSSStarHackParser, self).parse_declaration(tokens)
...         declaration.has_star_hack = has_star_hack
...         return declaration
...
>>> stylesheet = CSSStarHackParser().parse_stylesheet(css)
>>> stylesheet.errors
[]
>>> [(d.name, d.has_star_hack) for d in stylesheet.rules[0].declarations]
[('width', False), ('width', True)]
```

This class extends the `parse_declaration()` method. It removes any `*` delimeter `Token` at the start of a declaration, and adds a `has_star_hack` boolean attribute on parsed `Declaration` objects: `True` if a `*` was removed, `False` for "normal" declarations.

Parser methods

In addition to methods of the user API (see *Parsing a stylesheet*), here are the methods of the CSS 2.1 parser that can be overriden or extended:

Unparsed at-rules

Parsing helper functions

The *`tinycss.parsing`* module contains helper functions for parsing tokens into a more structured form:

Hacking tinycss

Bugs and feature requests

Bug reports, feature requests and other issues should got to the tinycss issue tracker on Github. Any suggestion or feedback is welcome. Please include in full any error message, trackback or other detail that could be helpful.

Installing the development version

First, get the latest git version:

```
git clone https://github.com/SimonSapin/tinycss.git
cd tinycss
```

You will need Cython and pytest. Installing in a virtualenv is recommended:

```
virtualenv env
. env/bin/activate
pip install Cython pytest
```

Then, install tinycss in-place with pip's *editable mode*. This will also build the accelerators:

```
pip install -e .
```

Running the test suite

Once you have everything installed (see above), just run pytest from the *tinycss* directory:

```
py.test
```

If the accelerators are not available for some reason, use the `TINYCSS_SKIP_SPEEDUPS_TESTS` environment variable:

```
TINYCSS_SKIP_SPEEDUPS_TESTS=1 py.test
```

If you get test failures on a fresh git clone, something may have gone wrong during the installation. Otherwise, you probably found a bug. Please *report it*.

Test in multiple Python versions with tox

tox automatically creates virtualenvs for various Python versions and runs the test suite there:

```
pip install tox
```

Change to the project's root directory and just run:

```
tox
```

tinycss comes with a pre-configured `tox.ini` file to test in CPython 2.6, 2.7, 3.1 and 3.2 as well as PyPy. You can change that with the `-e` parameter:

```
tox -e py27,py32
```

If you use `--` in the arguments passed to tox, further arguments are passed to the underlying `py.test` command:

```
tox -- -x --pdb
```

Building the documentation

This documentation is made with Sphinx:

```
pip install Sphinx
```

To build the HTML version of the documentation, change to the project's root directory and run:

```
python setup.py build_sphinx
```

The built HTML files are in `docs/_build/html`.

Making a patch and a pull request

If you would like to see something included in tinycss, please fork the repository on Github and make a pull request. Make sure to include tests for your change.

Mailing-list

tinycss does not have a mailing-list of its own for now, but the WeasyPrint mailing-list is appropriate to discuss it.

tinycss changelog

Version 0.4

Released on 2016-09-23.

- Add an __eq__ operator to Token object.
- Support Fonts 3.

Version 0.3

Released on 2012-09-18.

- Fix a bug when parsing 5c (an escaped antislash.)

Version 0.2

Released on 2012-04-27.

Breaking changes:

- Remove the `selectors3` module. The functionality has moved to the cssselect project.
- Simplify the API for `make_parser()`.

Version 0.1.1

Released on 2012-04-06.

Bug fixes:

- Error handling on exepected end of stylesheet in an at-rule head
- Fix the installation on ASCII-only locales

Version 0.1

Released on 2012-04-05.

First release. Parser support for CSS 2.1, Seloctors 3, Color 3 and Paged Media 3.

Python Module Index

t

Index

T