dict2css

Release 0.3.0.post1

A µ-library for constructing cascading style sheets from Python dictionaries.

Dominic Davis-Foster

Nov 22, 2023

Contents

`dict2css` provides an API similar to the `json` and toml modules, with *`dump()`* and *`load()`* functions. The *`dump()`* function takes a mapping of CSS selectors to mappings of CSS properties. Each property value may, optionally, be a two-element tuple containing the value and the string "important". The *`load()`* function returns a mapping with the same structure.

Chapter

ONE

Installation

1.1 from PyPI

```
$ python3 -m pip install dict2css --user
```

1.2 from Anaconda

First add the required channels

```
$ conda config --add channels https://conda.anaconda.org/conda-forge
$ conda config --add channels https://conda.anaconda.org/domdfcoding
```

Then install

```
$ conda install dict2css
```

1.3 from GitHub

```
$ python3 -m pip install git+https://github.com/sphinx-toolbox/dict2css@master --user
```

Chapter

TWO

dict2css

A µ-library for constructing cascasing style sheets from Python dictionaries.

See also: css-parser, which this library builds upon.

Data:

IMPORTANT	The string `'important'`.
Style	Type annotation representing a style for *make_style()* and *dumps()*.

Functions:

dumps(styles, *[, indent, ...])	Construct a cascading style sheet from a dictionary.
dump(styles, fp, *[, indent, ...])	Construct a style sheet from a dictionary and write it to `fp`.
loads(styles)	Parse a style sheet and return its dictionary representation.
load(fp)	Parse a cascading style sheet from the given file and return its dictionary representation.
make_style(selector, styles)	Create a CSS Style Rule from a dictionary.

Classes:

StyleSheet()	Represents a CSS style sheet.

IMPORTANT = 'important'

Type: `str`

The string `'important'`.

Style

Type annotation representing a style for *make_style()* and *dumps()*.

The keys are CSS properties.

The values can be either:

- A `str`, `float` or `None`, giving the value of the property.
- A `tuple` of the property's value (as above) and the priority such as *IMPORTANT* (which sets `!important` on the property).

Alias of `Mapping[str, Union[Sequence, str, int, None]]`

dumps(*styles*, *, *indent='\t'*, *trailing_semicolon=False*, *indent_closing_brace=False*, *minify=False*)

Construct a cascading style sheet from a dictionary.

`styles` is a mapping of CSS selector strings to styles, which map property names to their values:

```
styles = {".wy-nav-content": {"max-width": (px(1200), IMPORTANT)}}
print(dumps(styles))
```

```
.wy-nav-content {
    max-width: 1200px !important
}
```

See the *Style* object for more information on the layout.

The keys can also be media at-rules, with the values mappings of property names to their values:

```
styles = {
    "@media screen and (min-width: 870px)": {
        ".wy-nav-content": {"max-width": (px(1200), IMPORTANT)},
        },
    }
print(dumps(styles))
```

```
@media screen and (min-width: 870px) {
    .wy-nav-content {
        max-width: 1200px !important
    }
}
```

Parameters

- **styles** (Mapping[str, Union[Mapping[str, Union[Sequence, str, int, None]], Mapping]]) – A mapping of CSS selectors to styles.
- **indent** (str) – The indent to use, such as a tab (\t), two spaces or four spaces. Default '\t'.
- **trailing_semicolon** (bool) – Whether to add a semicolon to the end of the final property. Default False.
- **indent_closing_brace** (bool) – Default False.
- **minify** (bool) – Minify the CSS. Overrides all other options. Default False.

Return type str

Returns The style sheet as a string.

Changed in version 0.2.0: Added support for media at-rules.

dump(*styles*, *fp*, *, *indent='\t'*, *trailing_semicolon=False*, *indent_closing_brace=False*, *minify=False*)

Construct a style sheet from a dictionary and write it to fp.

```
styles = {".wy-nav-content": {"max-width": (px(1200), IMPORTANT)}}
dump(styles, ...)
```

```
.wy-nav-content {
    max-width: 1200px !important
}
```

See the *Style* object for more information on the layout.

The keys can also be media at-rules, with the values mappings of property names to their values:

```
styles = {
    "@media screen and (min-width: 870px)": {
        ".wy-nav-content": {"max-width": (px(1200), IMPORTANT)},
        },
    }
dump(styles, ...)
```

```
@media screen and (min-width: 870px) {
    .wy-nav-content {
        max-width: 1200px !important
    }
}
```

Parameters

- **styles** (Mapping[str, Union[Mapping[str, Union[Sequence, str, int, None]], Mapping]]) – A mapping of CSS selectors to styles.
- **fp** (Union[str, Path, PathLike, IO]) – An open file handle, or the filename of a file to write to.
- **indent** (str) – The indent to use, such as a tab (\t), two spaces or four spaces. Default '\t'.
- **trailing_semicolon** (bool) – Whether to add a semicolon to the end of the final property. Default False.
- **indent_closing_brace** (bool) – Default False.
- **minify** (bool) – Minify the CSS. Overrides all other options. Default False.

Changed in version 0.2.0:

- fp now accepts domdf_python_tools.typing.PathLike objects, representing the path of a file to write to.
- Added support for media at-rules.

loads(*styles*)

Parse a style sheet and return its dictionary representation.

New in version 0.2.0.

Parameters **styles** (str)

Return type MutableMapping[str, MutableMapping[str, Any]]

Returns The style sheet as a dictionary.

load(*fp*)

Parse a cascading style sheet from the given file and return its dictionary representation.

New in version 0.2.0.

Parameters **fp** (Union[str, Path, PathLike, IO]) – An open file handle, or the filename of a file to write to.

Return type MutableMapping[str, MutableMapping[str, Any]]

Returns The style sheet as a dictionary.

class StyleSheet

Represents a CSS style sheet.

Methods:

add(rule)	Add the `rule` to the style sheet.
add_style(selector, styles)	Add a style to the style sheet.
add_media_styles(media_query, styles)	Add a set of styles for a media query to the style sheet.
tostring()	Returns the style sheet as a string.

add(*rule*)

Add the `rule` to the style sheet.

Parameters **rule** (`css_parser.css.CSSRule`)

Return type `int`

add_style(*selector*, *styles*)

Add a style to the style sheet.

Parameters

- **selector** (`str`)
- **styles** (`Mapping[str, Union[Sequence, str, int, None]]`)

add_media_styles(*media_query*, *styles*)

Add a set of styles for a media query to the style sheet.

New in version 0.2.0.

Parameters

- **media_query** (`str`)
- **styles** (`Mapping[str, Mapping[str, Union[Sequence, str, int, None]]]`)

tostring()

Returns the style sheet as a string.

Return type `str`

make_style(*selector*, *styles*)

Create a CSS Style Rule from a dictionary.

Parameters

- **selector** (`str`)
- **styles** (`Mapping[str, Union[Sequence, str, int, None]]`)

Return type `css_parser.css.CSSStyleRule`

Chapter

THREE

dict2css.helpers

Helper functions.

New in version 0.2.0.

Functions:

em(val)	Helper function to format a number as a value in em.
px(val)	Helper function to format a number as a value in pixels.
rem(val)	Helper function to format a number as a value in rem.

em(*val*)

Helper function to format a number as a value in em.

Parameters **val** (Union[int, float, str])

Return type str

px(*val*)

Helper function to format a number as a value in pixels.

Parameters **val** (Union[int, float, str])

Return type str

rem(*val*)

Helper function to format a number as a value in rem.

Parameters **val** (Union[int, float, str])

Return type str

Chapter

FOUR

dict2css.serializer

Serializer for cascading style sheets.

New in version 0.2.0.

Classes:

CSSSerializer(*[, indent, ...])	Serializes a *StyleSheet* and its parts.

class CSSSerializer(*, *indent='\t'*, *trailing_semicolon=False*, *indent_closing_brace=False*, *minify=False*)

Serializes a *StyleSheet* and its parts.

This controls the formatting of the style sheet.

Parameters

- **indent** (str) – The indent to use, such as a tab (\t), two spaces or four spaces. Default '\t'.
- **trailing_semicolon** (bool) – Whether to add a semicolon to the end of the final property. Default False.
- **indent_closing_brace** (bool) – Default False.
- **minify** (bool) – Minify the CSS. Overrides all other options. Default False.

Methods:

reset_style()	Reset the serializer to its default style.
use()	Contextmanager to use this serializer for the scope of the with block.

reset_style()

Reset the serializer to its default style.

use()

Contextmanager to use this serializer for the scope of the with block.

Return type Iterator

Chapter

FIVE

Changelog

0.2.2

Changed the build backend from setuptools to whey.

0.2.1

Import `Iterator` from `typing` rather than from `collections`.

0.2.0

`dict2css.dumps()`

Added support for media at-rules.

`dict2css.dump()`

- `fp` now accepts `domdf_python_tools.typing.PathLike` objects, representing the path of a file to write to.
- Added support for media at-rules.

Additions

Functions

- *`dict2css.loads()`*
- *`dict2css.load()`*

Methods

- *`dict2css.StyleSheet.add_media_styles()`*

Modules

- *`dict2css.helpers`*
- *`dict2css.serializer`*

0.1.0

Initial release.

Python Module Index

d

Index