
dict2css

Release 0.3.0.post1

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

Dominic Davis-Foster

May 15, 2024

Contents

1	Installation	3
1.1	from PyPI	3
1.2	from Anaconda	3
1.3	from GitHub	3
2	dict2css	5
2.1	IMPORTANT	5
2.2	Style	5
2.3	dumps	5
2.4	dump	6
2.5	loads	7
2.6	load	7
2.7	StyleSheet	7
2.8	make_style	8
3	dict2css.helpers	9
3.1	em	9
3.2	px	9
3.3	rem	9
4	dict2css.serializer	11
4.1	CSSSerializer	11
5	Changelog	13
5.1	0.2.2	13
5.2	0.2.1	13
5.3	0.2.0	13
5.7	0.1.0	13
	Python Module Index	15
	Index	17

`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.

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
```


dict2css

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

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

Data:

<code>IMPORTANT</code>	The string 'important'.
<code>Style</code>	Type annotation representing a style for <code>make_style()</code> and <code>dumps()</code> .

Functions:

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

Classes:

<code>StyleSheet()</code>	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='\n'`, `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:

<code>add(rule)</code>	Add the <code>rule</code> to the style sheet.
<code>add_style(selector, styles)</code>	Add a style to the style sheet.
<code>add_media_styles(media_query, styles)</code>	Add a set of styles for a media query to the style sheet.
<code>tostring()</code>	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`

`dict2css.helpers`

Helper functions.

New in version 0.2.0.

Functions:

<code>em(val)</code>	Helper function to format a number as a value in em.
<code>px(val)</code>	Helper function to format a number as a value in pixels.
<code>rem(val)</code>	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`

dict2css.serializer

Serializer for cascading style sheets.

New in version 0.2.0.

Classes:

<code>CSSSerializer(*[, indent, ...])</code>	Serializes a <i>StyleSheet</i> 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:

<code>reset_style()</code>	Reset the serializer to its default style.
<code>use()</code>	Contextmanager to use this serializer for the scope of the <code>with</code> 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`

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

`dict2css`, [5](#)
`dict2css.helpers`, [9](#)
`dict2css.serializer`, [11](#)

A

`add()` (*StyleSheet method*), 8
`add_media_styles()` (*StyleSheet method*), 8
`add_style()` (*StyleSheet method*), 8

C

`CSSSerializer` (*class in dict2css.serializer*), 11

D

`dict2css`
 module, 5
`dict2css.helpers`
 module, 9
`dict2css.serializer`
 module, 11
`dump()` (*in module dict2css*), 6
`dumps()` (*in module dict2css*), 5

E

`em()` (*in module dict2css.helpers*), 9

I

`IMPORTANT` (*in module dict2css*), 5

L

`load()` (*in module dict2css*), 7
`loads()` (*in module dict2css*), 7

M

`make_style()` (*in module dict2css*), 8
`module`
 dict2css, 5
 dict2css.helpers, 9
 dict2css.serializer, 11

P

`px()` (*in module dict2css.helpers*), 9

R

`rem()` (*in module dict2css.helpers*), 9
`reset_style()` (*CSSSerializer method*), 11

S

`Style` (*in module dict2css*), 5
`StyleSheet` (*class in dict2css*), 7

T

`tostring()` (*StyleSheet method*), 8

U

`use()` (*CSSSerializer method*), 11