webcolors Documentation

Release 1.4

James Bennett

March 17, 2015

Contents

This module provides utility functions for working with the color names and color value formats defined by the HTML and CSS specifications for use in documents on the Web.

The home page, and development repository, of this module is <http://bitbucket.org/ubernostrum/webcolors/>. It can be installed by downloading a package from the "Downloads" tab there (and then running `setup.py install` from the package), or by downloading from the Python Package Index (<http://pypi.python.org/pypi/webcolors/>). Automated tools such as `easy_install` and `pip` can also be used (`easy_install webcolors` or `pip install webcolors`).

CHAPTER 1

An overview of colors on the Web

Colors on the Web are specified in the sRGB color space, where each color is made up of a red component, a green component and a blue component. This maps (fairly) cleanly to the red, green and blue components of pixels on a computer display, and to the cone cells of a human eye, which come in three sets roughly corresponding to the wavelengths of light associated with red, green and blue.

The HTML 4 specification defined two ways to specify sRGB colors:

- A hash mark ('#') followed by three pairs of hexdecimal digits, specifying values for red, green and blue components in that order; for example, `#0099cc`. Since each pair of hexadecimal digits can express 256 different values, this allows up to 256**3 or 16,777,216 unique colors to be specified (though, due to differences in display technology, not all of these colors may be clearly distinguished on any given physical display).
- A set of predefined color names which correspond to specific hexadecimal values; for example, `blue`. HTML 4 defines sixteen such colors.

In its description of color units, the CSS 1 specification added three new ways to specify sRGB colors:

- A hash mark followed by three hexadecimal digits, which is expanded into three hexadecimal pairs by repeating each digit; thus `#09c` is equivalent to `#0099cc`.
- The string "rgb", followed by parentheses, between which are three base-10 integers each between 0 and 255, inclusive, which are taken to be the values of the red, green and blue components in that order; for example, `rgb(0, 153, 204)`.
- The same as above, except using percentages instead of numeric values; for example, `rgb(0%, 60%, 80%)`.

CSS 1 also suggested a set of sixteen color names. These names were identical to the set defined in HTML 4, but CSS 1 did not provide definitions of their values, and stated that they were taken from "the Windows VGA palette".

In its section on colors, the CSS 2 specification allowed the same methods of specifying colors as CSS 1, and defined and provided values for sixteen named colors, identical to the set found in HTML 4.

The CSS 2.1 revision did not add any new methods of specifying sRGB colors, but did define one additional named color: `orange`.

The CSS 3 color module (currently a W3C Working Draft) adds one new way to specify colors:

- A hue-saturation-lightness triple (HSL), using the construct `hsl()`.

CSS 3 also adds support for variable opacity of colors, by allowing the specification of alpha-channel information through the `rgba()` and `hsla()` constructs. These are used similarly to the `rgb()` and `hsl()` constructs, except a fourth value is supplied indicating the level of opacity from `0.0` (completely transparent) to `1.0` (completely opaque). Though not technically a color, the keyword `transparent` is also made available in lieu of a color value, and corresponds to `rgba(0,0,0,0)`.

Finally, CSS 3 defines a new set of color names. This set is taken directly from the named colors defined for SVG (Scalable Vector Graphics) markup, and is a proper superset of the named colors defined in CSS 2.1. This set also has significant overlap with traditional X11 color sets as defined by the `rgb.txt` file on many Unix and Unix-like operating systems, though the correspondence is not exact: the set of X11 colors is not standardized, and the set of CSS 3/SVG colors contains some definitions which diverge significantly from customary X11 definitions (for example, CSS 3/SVG's `green` is not equivalent to X11's `green`; the color which X11 designates `green` is designated `lime` in CSS 3/SVG).

CHAPTER 2

What this module supports

This module supports the following methods of specifying sRGB colors, and conversions between them:

- Six-digit hexadecimal.
- Three-digit hexadecimal.
- Integer `rgb()` triplet.
- Percentage `rgb()` triplet.
- Varying selections of predefined color names.

This module does not support `hsl()` triplets, nor does it support opacity/alpha-channel information via `rgba()` or `hsla()`.

If you need to convert between RGB-specified colors and HSL-specified colors, or colors specified via other means, consult the colorsys module in the Python standard library, which can perform conversions amongst several common color systems.

2.1 Normalization and conventions

For colors specified via hexadecimal values, this module will accept input in the following formats:

- A hash mark (#) followed by three hexadecimal digits, where digits A-F may be upper- or lower-case.
- A hash mark (#) followed by six hexadecimal digits, where digits A-F may be upper- or lower-case.

For output which consists of a color specified via hexadecimal values, and for functions which perform intermediate conversion to hexadecimal before returning a result in another format, this module always normalizes such values to the following format:

- A hash mark (#) followed by six hexadecimal digits, with digits A-F forced to lower-case.

The function `normalize_hex()` in this module can be used to perform this normalization manually if desired.

For colors specified via predefined names, this module will accept input in the following formats:

- An entirely lower-case name, such as `aliceblue`.
- A name using CamelCase, such as `AliceBlue`.

For output which consists of a color specified via a predefined name, and for functions which perform intermediate conversion to a predefined name before returning a result in another format, this module always normalizes such values to be entirely lower-case.

For colors specified via `rgb()` triplets, values contained in the triplets will be normalized via clipping in accordance with CSS:

- Integer values less than 0 will be normalized to 0, and percentage values less than 0% will be normalized to 0%.
- Integer values greater than 255 will be normalized to 255, and percentage values greater than 100% will be normalized to 100%.

The functions `normalize_integer_triplet()` and `normalize_percent_triplet()` in this module can be used to perform this normalization manually if desired.

For purposes of identifying the specification from which to draw the selection of defined color names, this module recognizes the following identifiers:

`html4` The HTML 4 named colors.

`css2` The CSS 2 named colors.

`css21` The CSS 2.1 named colors.

`css3` The CSS 3/SVG named colors.

The CSS 1 specification is not represented here, as it merely "suggested" a set of color names, and declined to provide values for them.

CHAPTER 3

Module contents

webcolors.**normalize_hex**(*hex_value*)

Normalize a hexadecimal color value to the following form and return the result:

```
#[a-f0-9]{6}
```

In other words, the following transformations are applied as needed:

- If the value contains only three hexadecimal digits, it is expanded to six.
- The value is normalized to lower-case.

If the supplied value cannot be interpreted as a hexadecimal color value, `ValueError` is raised. See *the conventions used by this module* for information on acceptable formats for hexadecimal values.

Examples:

```
>>> normalize_hex('#0099cc')
'#0099cc'
>>> normalize_hex('#0099CC')
'#0099cc'
>>> normalize_hex('#09c')
'#0099cc'
>>> normalize_hex('#09C')
'#0099cc'
>>> normalize_hex('0099cc')
Traceback (most recent call last):
    ...
ValueError: '0099cc' is not a valid hexadecimal color value.
```

Parameters **hex_value** (*str*) – The hexadecimal color value to normalize.

Return type str

webcolors.**normalize_integer_triplet**(*rgb_triplet*)

Normalize an integer `rgb()` triplet so that all values are within the range 0-255 inclusive.

Examples:

```
>>> normalize_integer_triplet((128, 128, 128))
(128, 128, 128)
>>> normalize_integer_triplet((0, 0, 0))
(0, 0, 0)
>>> normalize_integer_triplet((255, 255, 255))
(255, 255, 255)
```

```
>>> normalize_integer_triplet((270, -20, 128))
(255, 0, 128)
```

Parameters **rgb_triplet** (*3-tuple of integers*) – The integer `rgb()` triplet to normalize.

Return type 3-tuple of integers

`webcolors.`**`normalize_percent_triplet`**(*rgb_triplet*)

Normalize a percentage `rgb()` triplet to that all values are within the range 0%-100% inclusive.

Examples:

```
>>> normalize_percent_triplet(('50%', '50%', '50%'))
('50%', '50%', '50%')
>>> normalize_percent_triplet(('0%', '100%', '0%'))
('0%', '100%', '0%')
>>> normalize_percent_triplet(('-10%', '250%', '500%'))
('0%', '100%', '100%')
```

Parameters **rgb_triplet** (*3-tuple of strings*) – The percentage `rgb()` triplet to normalize.

Return type 3-tuple of strings

3.1 Constants

`webcolors.`**`html4_names_to_hex`**

A dictionary whose keys are the names of the defined colors in HTML 4 (normalized to lowercase), and whose values are the corresponding (normalized) hexadecimal color values.

`webcolors.`**`html4_hex_to_names`**

A dictionary whose keys are (normalized) hexadecimal color values of the named HTML 4 colors, and whose values are the corresponding (normalized to lowercase) names.

`webcolors.`**`css2_names_to_hex`**

A dictionary whose keys are the names of the defined colors in CSS 2 (normalized to lowercase), and whose values are the corresponding (normalized) hexadecimal color values.

Because CSS 2 defines the same set of colors as HTML 4, this is simply an alias for `html4_names_to_hex`.

`webcolors.`**`css2_hex_to_names`**

A dictionary whose keys are (normalized) hexadecimal color values of the named CSS 2 colors, and whose values are the corresponding (normalized to lowercase) names.

Because CSS 2 defines the same set of colors as HTML 4, this is simply an alias for `html4_hex_to_names`.

`webcolors.`**`css21_names_to_hex`**

A dictionary whose keys are the names of the defined colors in CSS 2.1 (normalized to lowercase), and whose values are the corresponding (normalized) hexadecimal color values.

`webcolors.`**`css21_hex_to_names`**

A dictionary whose keys are the (normalized) hexadecimal color values of the named CSS 2.1 colors, and whose values are the corresponding (normalized to lowercase) names.

`webcolors.`**`css3_names_to_hex`**

A dictionary whose keys are the names of the defined colors in the CSS 3 color module (normalized to lowercase), and whose values are the corresponding (normalized) hexadecimal color values.

webcolors.**css3_hex_to_names**
A dictionary whose keys are the (normalized) hexadecimal color values of the named CSS 3 colors, and whose values are the corresponding (normalized to lowercase) names.

3.2 Conversions from color names to other formats

webcolors.**name_to_hex**(*name*, *spec='css3'*)
Convert a color name to a normalized hexadecimal color value.

The color name will be normalized to lower-case before being looked up, and when no color of that name exists in the given specification, `ValueError` is raised.

Examples:

```
>>> name_to_hex('white')
'#ffffff'
>>> name_to_hex('navy')
'#000080'
>>> name_to_hex('goldenrod')
'#daa520'
>>> name_to_hex('goldenrod', spec='html4')
Traceback (most recent call last):
    ...
ValueError: 'goldenrod' is not defined as a named color in html4.
>>> name_to_hex('goldenrod', spec='css5')
Traceback (most recent call last):
    ...
TypeError: 'css5' is not a supported specification for color name lookups; supported specificati
```

Parameters

- **name** (*str*) – The color name to convert.
- **spec** (*str*) – The specification from which to draw the list of color names; valid values are `html4`, `css2`, `css21` and `css3`. Default is `css3`.

Return type str

webcolors.**name_to_rgb**(*name*, *spec='css3'*)
Convert a color name to a 3-tuple of integers suitable for use in an `rgb()` triplet specifying that color.

The color name will be normalized to lower-case before being looked up, and when no color of that name exists in the given specification, `ValueError` is raised.

Examples:

```
>>> name_to_rgb('white')
(255, 255, 255)
>>> name_to_rgb('navy')
(0, 0, 128)
>>> name_to_rgb('goldenrod')
(218, 165, 32)
```

Parameters

- **name** (*str*) – The color name to convert.
- **spec** (*str*) – The specification from which to draw the list of color names; valid values are `html4`, `css2`, `css21` and `css3`. Default is `css3`.

Return type 3-tuple of integers

webcolors.**name_to_rgb_percent**(*name*, *spec='css3'*)

Convert a color name to a 3-tuple of percentages (as strings) suitable for use in an `rgb()` triplet specifying that color.

The color name will be normalized to lower-case before being looked up, and when no color of that name exists in the given specification, `ValueError` is raised.

Examples:

```
>>> name_to_rgb_percent('white')
('100%', '100%', '100%')
>>> name_to_rgb_percent('navy')
('0%', '0%', '50%')
>>> name_to_rgb_percent('goldenrod')
('85.49%', '64.71%', '12.5%')
```

Parameters

- **name** (*str*) – The color name to convert.
- **spec** (*str*) – The specification from which to draw the list of color names; valid values are `html4`, `css2`, `css21` and `css3`. Default is `css3`.

Return type 3-tuple of strings

3.3 Conversions from hexadecimal color values to other formats

webcolors.**hex_to_name**(*hex_value*, *spec='css3'*)

Convert a hexadecimal color value to its corresponding normalized color name, if any such name exists.

The hexadecimal value will be normalized before being looked up, and when no color name for the value is found in the given specification, `ValueError` is raised.

Examples:

```
>>> hex_to_name('#ffffff')
'white'
>>> hex_to_name('#fff')
'white'
>>> hex_to_name('#000080')
'navy'
>>> hex_to_name('#daa520')
'goldenrod'
>>> hex_to_name('#daa520', spec='html4')
Traceback (most recent call last):
    ...
ValueError: '#daa520' has no defined color name in html4.
>>> hex_to_name('#daa520', spec='css5')
Traceback (most recent call last):
    ...
TypeError: 'css5' is not a supported specification for color name lookups; supported specificati
```

Parameters

- **hex_value** (*str*) – The hexadecimal color value to convert.

- **spec** (*str*) – The specification from which to draw the list of color names; valid values are `html4`, `css2`, `css21` and `css3`. Default is `css3`.

Return type str

`webcolors.`**`hex_to_rgb`**(*hex_value*)

Convert a hexadecimal color value to a 3-tuple of integers suitable for use in an `rgb()` triplet specifying that color.

The hexadecimal value will be normalized before being converted.

Examples:

```
>>> hex_to_rgb('#fff')
(255, 255, 255)
>>> hex_to_rgb('#000080')
(0, 0, 128)
```

Parameters **hex_value** (*str*) – The hexadecimal color value to convert.

Return type 3-tuple of integers

`webcolors.`**`hex_to_rgb_percent`**(*hex_value*)

Convert a hexadecimal color value to a 3-tuple of percentages (as strings) suitable for use in an `rgb()` triplet representing that color.

The hexadecimal value will be normalized before converting.

Examples:

```
>>> hex_to_rgb_percent('#ffffff')
('100%', '100%', '100%')
>>> hex_to_rgb_percent('#000080')
('0%', '0%', '50%')
```

Parameters **hex_value** (*str*) – The hexadecimal color value to convert.

Return type 3-tuple of strings

3.4 Conversions from integer `rgb()` triplets to other formats

`webcolors.`**`rgb_to_name`**(*rgb_triplet*, *spec='css3'*)

Convert a 3-tuple of integers, suitable for use in an `rgb()` color triplet, to its corresponding normalized color name, if any such name exists.

To determine the name, the triplet will be converted to a normalized hexadecimal value. When no corresponding name for the value is found in the given specification, `ValueError` is raised.

Examples:

```
>>> rgb_to_name((255, 255, 255))
'white'
>>> rgb_to_name((0, 0, 128))
'navy'
```

Parameters

- **rgb_triplet** (*3-tuple of integers*) – The `rgb()` triplet

- **spec** (*str*) – The specification from which to draw the list of color names; valid values are `html4`, `css2`, `css21` and `css3`. Default is `css3`.

Return type str

`webcolors.`**`rgb_to_hex`**(*rgb_triplet*)

Convert a 3-tuple of integers, suitable for use in an `rgb()` color triplet, to a normalized hexadecimal value for that color.

Examples:

```
>>> rgb_to_hex((255, 255, 255))
'#ffffff'
>>> rgb_to_hex((0, 0, 128))
'#000080'
```

Parameters **rgb_triplet** (*3-tuple of integers*) – The `rgb()` triplet.

Return type str

`webcolors.`**`rgb_to_rgb_percent`**(*rgb_triplet*)

Convert a 3-tuple of integers, suitable for use in an `rgb()` color triplet, to a 3-tuple of percentages (as strings) suitable for use in representing that color.

This function makes some trade-offs in terms of the accuracy of the final representation; for some common integer values, special-case logic is used to ensure a precise result (e.g., integer 128 will always convert to '50%', integer 32 will always convert to '12.5%'), but for all other values a standard Python `float` is used and rounded to two decimal places, which may result in a loss of precision for some values.

Examples:

```
>>> rgb_to_rgb_percent((255, 255, 255))
('100%', '100%', '100%')
>>> rgb_to_rgb_percent((0, 0, 128))
('0%', '0%', '50%')
>>> rgb_to_rgb_percent((218, 165, 32))
('85.49%', '64.71%', '12.5%')
```

Parameters **rgb_triplet** (*3-tuple of integers*) – The `rgb()` triplet.

Return type 3-tuple of strings

3.5 Conversions from percentage `rgb()` triplets to other formats

`webcolors.`**`rgb_percent_to_name`**(*rgb_percent_triplet*, *spec='css3'*)

Convert a 3-tuple of percentages, suitable for use in an `rgb()` color triplet, to its corresponding normalized color name, if any such name exists.

To determine the name, the triplet will be converted to a normalized hexadecimal value. When no corresponding name for the value is found in the given specification, `ValueError` is raised.

Examples:

```
>>> rgb_percent_to_name(('100%', '100%', '100%'))
'white'
>>> rgb_percent_to_name(('0%', '0%', '50%'))
'navy'
```

```
>>> rgb_percent_to_name(('85.49%', '64.71%', '12.5%'))
'goldenrod'
```

Parameters

- **rgb_percent_triplet** (*3-tuple of strings*) – The `rgb()` triplet.
- **spec** (*str*) – The specification from which to draw the list of color names; valid values are `html4`, `css2`, `css21` and `css3`. Default is `css3`.

Return type str

webcolors.**rgb_percent_to_hex**(*rgb_percent_triplet*)

Convert a 3-tuple of percentages, suitable for use in an `rgb()` color triplet, to a normalized hexadecimal color value for that color.

Examples:

```
>>> rgb_percent_to_hex(('100%', '100%', '0%'))
'#ffff00'
>>> rgb_percent_to_hex(('0%', '0%', '50%'))
'#000080'
>>> rgb_percent_to_hex(('85.49%', '64.71%', '12.5%'))
'#daa520'
```

Parameters **rgb_percent_triplet** (*3-tuple of strings*) – The `rgb()` triplet.

Return type str

webcolors.**rgb_percent_to_rgb**(*rgb_percent_triplet*)

Convert a 3-tuple of percentages, suitable for use in an `rgb()` color triplet, to a 3-tuple of integers suitable for use in representing that color.

Some precision may be lost in this conversion. See the note regarding precision for `rgb_to_rgb_percent()` for details. Generally speaking, the following is true for any 3-tuple `t` of integers in the range 0...255 inclusive:

```
t == rgb_percent_to_rgb(rgb_to_rgb_percent(t))
```

Examples:

```
>>> rgb_percent_to_rgb(('100%', '100%', '100%'))
(255, 255, 255)
>>> rgb_percent_to_rgb(('0%', '0%', '50%'))
(0, 0, 128)
>>> rgb_percent_to_rgb(('85.49%', '64.71%', '12.5%'))
(218, 165, 32)
```

Parameters **rgb_percent_triplet** (*3-tuple of strings*) – The `rgb()` triplet.

Return type 3-tuple of integers

CHAPTER 4

Known issues

Due to the use of floats to handle percentages, some precision may be lost when converting to or from percentage `rgb()` triplets; see `rgb_to_rgb_percent()` and `rgb_percent_to_rgb()` for details.

Bugs should be reported to the issue tracker on Bitbucket.

W

C

H

N

R

W