design Documentation

Release 0.1.3

Audrey Roy

Sep 27, 2017

Contents

1 Design **3**

2 Installation **5**

3 Usage **7**
3.1 Still Pre-Alpha 7

4 Contributing **9**
4.1 Submitting Feedback 9
4.2 Getting Started 9
4.3 Pull Request Guidelines 10

5 Credits **11**
5.1 Development Lead 11
5.2 Contributors 11

6 History **13**
6.1 0.1.2 (2013-07-11) 13
6.2 0.1.1 (2013-07-11) 13
6.3 0.1.0 (2013-07-11) 13

7 design **15**
7.1 design Package 15

8 Indices and tables **17**

Python Module Index **19**

Contents:

CHAPTER 1

Design

Design is a command-line tool that generates various common web design elements: borders, patterns, textures, gradients, etc.

- Documentation: https://design.readthedocs.org
- GitHub: https://github.com/audreyr/design
- Free software: BSD license
- PyPI: https://pypi.python.org/pypi/design

CHAPTER 2

Installation

At the command line:

```
$ easy_install design
```

Or, if you have virtualenvwrapper installed:

```
$ mkvirtualenv design
$ pip install design
```

CHAPTER 3

Usage

Still Pre-Alpha

This isn't ready to use yet, but you could imagine something like this:

```
from colors import rgb
from design import buttons

buttons.make_arrow_button(
    width=120,
    height=30,
    color=rgb(255, 129, 190),
    push_depth=8,
    glow_on_hover=True
)
```

The above would result in these files being generated:

- img/arrow.png
- css/arrow.css (Or maybe a Compass file. Haven't decided yet.)
- js/arrow.js
- arrow.html (Demo of the arrow in action.)

Ideas/feedback? File an issue!

CHAPTER 4

Contributing

Contributions are welcome!

Submitting Feedback

The best way to send feedback is to file an issue at https://github.com/audreyr/design/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

Getting Started

Here's how to set up *design* for local development.

1. Fork the *design* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/design.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv design
$ cd design/
$ python setup.py develop
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

Now you can make your changes locally.

5. When you're done making changes, check that your changes pass the tests and flake8:

```
$ python -m unittest discover tests
$ flake8 design
$ flake8 tests
$ flake8 examples
```

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

Pull Request Guidelines

Before you submit a pull request:

1. TODO

CHAPTER 5

Credits

Development Lead

- Audrey Roy <audreyr@gmail.com>

Contributors

- Éric Araujo (@merwok)

CHAPTER 6

History

0.1.2 (2013-07-11)

- Cleanup. Fixes to pass tests.

0.1.1 (2013-07-11)

- Tiny setup.py fixes.

0.1.0 (2013-07-11)

- First release on PyPI.

CHAPTER 7

design

design Package

design Package

borders Module

borders

Functions for creating border pattern graphics.

design.borders.**circles**(*width=12*, *height=12*, *color=<RGBColor red: 255, green: 255, blue: 255>*)
Draws a repeatable circle border pattern.

design.borders.**circles_pil**(*width*, *height*, *color*)
Implementation of circle border with PIL.

design.borders.**circles_pycairo**(*width*, *height*, *color*)
Implementation of circle border with PyCairo.

clouds Module

clouds

Functions for creating cloud graphics.

design.clouds.**draw_circle**(*ctx*, *x*, *y*, *radius*, *cairo_color*)
Draw a circle. :param radius: radius in pixels :param cairo_color: normalized rgb color

design.clouds.**draw_cloud**(*width=140*, *height=60*, *color=<RGBColor red: 255, green: 255, blue: 255>*)
Draw a cloud with the given width, height, and color.

`gradients` Module

gradients

Functions for creating gradient graphics.

Note: CSS3 gradients are better than image gradient strips in most cases. See http://www.colorzilla.com/gradient-editor/

`design.gradients.`**`vertical_strip`**(*width=10*, *height=100*, *color=<RGBColor red: 100, green: 100, blue: 100>*, *subtlety=0.1*)

Draws a subtle vertical gradient strip.

`design.gradients.`**`vertical_white`**(*width=10*, *height=100*, *subtlety=0.1*)

Draws a subtle vertical gradient strip: white with varying alpha.

CHAPTER 8

Indices and tables

- genindex
- modindex
- search

Python Module Index

d

design.__init__, 15
design.borders, 15
design.clouds, 15
design.gradients, 16

Index

C

circles() (in module design.borders), 15
circles_pil() (in module design.borders), 15
circles_pycairo() (in module design.borders), 15

D

design.__init__ (module), 15
design.borders (module), 15
design.clouds (module), 15
design.gradients (module), 16
draw_circle() (in module design.clouds), 15
draw_cloud() (in module design.clouds), 15

V

vertical_strip() (in module design.gradients), 16
vertical_white() (in module design.gradients), 16