colosseum Documentation

Release stable

August 09, 2015

Contents

1 Quickstart 3

2 Community 5

3 Contributing 7

4 Acknowledgements 9

A (partial) implementation of the CSS box and flexbox layout algorithm.

The following CSS attributes and value types are supported:

Name	Value
width, height	positive number
min_width, min_height	positive number
max_width, max_height	positive number
left, right, top, bottom	number
margin, margin_left, margin_right, margin_top, margin_bottom	number
padding, padding_left, padding_right, padding_top, padding_bottom	positive number
border_width, border_left_width, border_right_width, border_top_width, border_bottom_width	positive number
flex_direction	`"column"`, `"row"`
justify_content	`"flex-start"`, `"center"`, `"flex-end"`, `"space-between"`, `"space-around"`
align_items, align_self	`"flex-start"`, `"center"`, `"flex-end"`, `"stretch"`
flex	positive number
flex_wrap	`"wrap"`, `"nowrap"`
position	`"relative"`, `"absolute"`

CHAPTER 1

Quickstart

In your virtualenv, install Colosseum:

```
$ pip install colosseum
```

Then, you can instantiate ‘‘CSSnode‘‘s, and query the layout that results:

```
>>> from colosseum import CSSNode, ROW, COLUMN
>>> node = CSSNode(width=1000, height=1000, flex_direction=ROW)
>>> node.children.add(CSSNode(width=100, height=200))
>>> node.children.add(CSSNode(width=300, height=150))
>>> layout = node.layout
>>> print(layout)
<Layout (1000x1000 @ 0,0)>
>>> layout.width
1000
>>> layout.height
1000
>>> layout.top
0
>>> layout.left
0
>>> for child in node.children:
...     print(child.layout)
<Layout (100x200 @ 0,0)>
<Layout (300x150 @ 100,0)>
```

Requesting the `layout` attribute of a `CSSNode` forces the box model to be evaluated. Once evaluated, the layout will be cached. Modifying any CSS property on a node will mark the layout as dirty, and the layout will be recomputed the next time the layout is accessed. For example, if we switch the outer node to be a “column” flex box, rather than a “row” flex box, you’ll see the coordinates of the child boxes update to reflect a vertical, rather than horizontal layout:

```
>>> node.flex_direction = COLUMN
>>> print(node.layout)
<Layout (1000x1000 @ 0,0)>
>>> for child in node.children:
...     print(child.layout)
<Layout (100x200 @ 0,0)>
<Layout (300x150 @ 0,200)>
```

Style attributes can also be set in bulk, using the `style()` method on a `CSSNode`:

```
>>> node.style(width=1500, height=800)
>>> print(node.layout)
<Layout (1500x800 @ 0,0)>
```

Style attributes can also be removed by deleting the attribute on the `CSSNode`. The value of the property will revert to the default:

```
>>> node.style(margin_top=10, margin_left=20)
>>> print(node.layout)
<Layout (1500x800 @ 20,10)>
>>> del(node.margin_left)
>>> print(node.margin_left)
0
>>> print(node.layout)
<Layout (1500x800 @ 0,10)>
```

Child nodes can also be defined declaratively by providing the child nodes as arguments to the parent node at time of construction. The original example could have been defined as follows:

```
>>> node = CSSNode(
...     CSSNode(width=100, height=200),
...     CSSNode(width=300, height=150),
...     width=1000,
...     height=1000,
...     flex_direction=ROW
... )
```

CHAPTER 2

Community

Colosseum is part of the BeeWare suite. You can talk to the community through:

- @pybeeware on Twitter
- The BeeWare Users Mailing list, for questions about how to use the BeeWare suite.
- The BeeWare Developers Mailing list, for discussing the development of new features in the BeeWare suite, and ideas for new tools for the suite.

CHAPTER 3

Contributing

If you experience problems with Colosseum, log them on GitHub. If you want to contribute code, please fork the code and submit a pull request.

CHAPTER 4

Acknowledgements

The algorithm and test suite for this library is a language port of CSS-layout project, open-sourced by Facebook.