# Jeet – Stylus <http://jeet.gs/>

### Installation

* Install [NodeJS](http://nodejs.org/)
* npm install -D jeet

### Usage Example

$ stylus -w style.styl

// style.styl

@import 'node\_modules/jeet/jeet'

debug()

section

center()

aside

column(1/3)

article

column(2/3)

<html>

<head>

<link rel="stylesheet" href="style.css">

</head>

<body>

<section>

<aside>Sidebar</aside>

<article>Content</article>

</section>

</body>

</html>

### Namespacing and Aliases

Jeet doesn't namespace or alias by default anymore (to keep the API simple/clean). If you need namespacing or aliasing, you can wrap mixins within namespace/alias mixins.

// Namespace Example

jeet-column()

column(arguments)

// Alias Example

col()

column(arguments)

Jeet's API.

Interact with our docs.

**COLUMN**

*($ratios: 1, $offset: 0, $cycle: 0, $gutter: map-get($jeet, 'gutter'))*

Specify an initial ratio, either as fractions or decimals, then pass the parent container's context ratio to maintain consistent gutters as you nest.

Specify a ratio to make your offset have a margin-left. Make it negative to give it a margin-right instead. e.g. column(1/4, $offset: 1/4)would create a column the quarter of the size of its container and push it to the right a quarter.

Understanding cycle is important. Want to make a gallery but don't want to have a <div class="row"> every 4 pictures? column(1/4, $cycle: 4) - done.

Want to change it up when you get down to mobile? Maybe just show 2 images per row? column(1/2, $cycle: 2) - done.

Need to adjust column gutters for a specific container? col(1/4, $gutter: .5). Note the gutter isn't a fixed width.

VIEW AN EXAMPLE

**COLUMN-WIDTH**

*($ratios: 1, $gutter: map-get($jeet, 'gutter'))*

A function to return strictly the column width with none of the styles.

**COLUMN-GUTTER**

*($ratios: 1, $gutter: map-get($jeet, 'gutter'))*

A function that returns the gutter size.

**SPAN**

*($ratio: 1, $offset: 0, $cycle: 0)*

Need a grid without the gutters? For instance, for a horizontal navigation where you want buttons touching. Do so with span(1/5). No need to pass more than one ratio since we don't need to worry about the math involved with gutters and all that.

VIEW AN EXAMPLE

**MOVE**

*($ratios: 0, $col-or-span: column, $gutter: map-get($jeet, 'gutter'))*

Source ordering works in Jeet by assigning position: relative and then a left offset equal to the ratio passed.

You can specify if this is a column or span movement to include gutters or not. This works similar to offset in that it can accept negative values to shift the other direction.

move can accept multiple context ratios to maintain perfect nested sizing. move also accepts custom gutter sizing, just make sure your gutter sizes match the gutter sizes of your original elements.

VIEW AN EXAMPLE

**UNMOVE**

*()*

Accepts no values. unmove() is a great helper function to quickly disable whatever source ordering you've done to an element.

**DEBUG**

*($color: blue, $important: false)*

Debug mode assigns a lightly translucent background to every element on the page. It's a little trick picked up over the years that makes visualizing the structure of your site trivial.

VIEW AN EXAMPLE

**CENTER**

*($max-width: map-get($jeet, 'max-width'), $pad: 0)*

This is a shortcut to easily center containers. The pad variable sets padding on the left and right.

VIEW AN EXAMPLE

**UNCENTER**

*()*

Resets styles associated with center().

**STACK**

*($pad: 0, $align: false)*

A helper mixin to "stack" elements on top of each other. Useful for mobile views. Accepts padding and/or text alignment.

VIEW AN EXAMPLE

**UNSTACK**

*()*

Cancel that stack(). This won't revert back to your column() calls. For that, manually call your column() method again.

**ALIGN**

*($direction: both)*

Aligning blocks relative to their container (so their parent container needs position: relative) with position: absolute and fancy positioning and transform. Vertical alignment is now trivial in IE9+ browsers.

VIEW AN EXAMPLE

**CLEARFIX**

*()*

A simple/kinda-modern clearfix. Use this to wrap any set of column()s or span()s. If you need super-old browser support, you can create your own clearfix mixin (use the clearfix namespace) with stuff like :beforeand \*zoom: 1 (look up an old clearfix).