Styling Maps Using CartoCSS

With CartoCSS you style a layer by setting properties on a layer's features. You do this by writing a series of statements. A statement takes the following form:

```
selector {
     property: value;
}
```

Use as many property-value pairs in a statement as is necessary.

Common properties

Markers (points)

`marker-fill`	inner part's color (color string)
`marker-fill-opacity`	inner part's opacity (0 to 1, lower is less visible)
`marker-line-color`	outer part's color
`marker-line-opacity`	outer part's opacity
`marker-height`	height (number, pixels)
`marker-width`	width (number, pixels)
`marker-allow-overlap`	draw all markers, even if they'll overlap (true/false)

Lines

`line-color`	color of line (color string)
`line-width`	width of line (number, pixels)
`line-opacity`	opacity of line (see marker-fill-opacity)

Polygons

`polygon-fill`	color of inside of polygon
`polygon-opacity`	opacity of inside of polygon

(Style the outside of polygons using `line-*` properties.)

See all properties in the official documentation: **http://bit.ly/cartocss-docs**

Advanced selectors

Selectors

You need to select a layer in order to style the features on that layer. In CartoDB, this is just the name of the table you are styling, followed by #. So if you uploaded a table called `mysecretlocations`, you could give all the markers on that layer a width of 3 using this statement:

```
#mysecretlocations {
     marker-width: 3;
}
```

Conditional selectors

Style by the **zoom level** of the map:

```
#layer-name[zoom >= 5] { ... }
```

Style features by their **attributes**:

```
#layer-name[attribute = value] { ... }
```

for example, if the attribute (column in CartoDB) is text:

```
#buildings[state = 'New York'] { ... }
```

If the column is a number:

```
#buildings[height > 50] { ... }
```

Use any of the following in your conditional selectors:

= (equal),

!= (not equal),

>= (greater than or equal),

<= (less than or equal),

> (greater than),

< (less than)

Combining selectors

You can **combine conditional selectors** by putting them right next to each other:

```
#layer-name[attr1 = value1][attr2 > value2] { ... }
```

This statement will only apply to features where **all** conditions are true. You can combine as many conditions as needed in this way. For example, to style buildings in New York over 50 feet tall, you might write:

```
#buildings[state = 'New York'][height > 50] { ... }
```

If you find yourself writing things like this to apply styles when one condition or the other is true (`attr1 = value1 OR attr2 > value2`):

```
#layer-name[attr1 = value1] {
     property: value;
}
#layer-name[attr2 > value2] {
     property: value;
}
```

consider separating the selectors with a comma:

```
#layer-name[attr1 = value1],
#layer-name[attr2 > value2] {
     property: value;
}
```

This does the same thing, but you don't have to repeat the styles (`property: value`) and if you have to change it later it will be faster.

Finally, you can **nest statements**. This says the same thing as the statement above:

```
#layer-name {
     [attr1 = value1],
     [attr2 > value2] {
          property: value;
     }
}
```

Let's make this more concrete:

```
#buildings {
     [state = 'New York'],
     [height > 50] {
          marker-fill: red;
     }
}
```

This styles features in the `buildings` layer that either have `state` set to New York or `height` greater than 50 such that their marker fill is red.

You will likely use multiple statements on one map:

```
#layer-name[zoom >= 5] { ... }
#layer-name[zoom >= 10] { ... }
#layer-name[zoom >= 15] { ... }
```

but it is equivalent and preferred that these statements are *nested*:

```
#layer-name {
     [zoom >= 5] { ... }
     [zoom >= 10] { ... }
     [zoom >= 15] { ... }
}
```

Variables

Sometimes you will find yourself repeating values in your statements. Your statements can be made more flexible using variables. Creating a variable looks like this:

```
@variable: value;
```

for example:

```
@roadcolor: #ff307a;
```

Then, instead of using the value in your statements, use `@variable`. For example:

```
#roads {
     line-color: @roadcolor;
}
```