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: **https://www.mapbox.com/carto/api/2.1.0/**

Advanced selectors

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:

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

Compare an attribute with a value using any of the following:

= (equal),

!= (not equal),

>= (greater than or equal),

<= (less than or equal),

> (greater than),

< (less than)

Combining selectors

You can combine conditional selectors:

```
#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.

Instead of writing

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

separate the selectors with a comma:

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

```
}
```

and it's nicer to nest the selectors like so:

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

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;
}
```