CartoCSS中文参考手册

前言

这是一份根据carto的最新文档和MapBox Studio的在线帮助文档综合整理翻译的中文版CartoCSS使用与语言参考手册。翻译的目的是为了方便越来越多的使用CartoCSS进行制图的中文用户了解这种制图样式语言中各种属性的含义及用法。读这份文档需要具有一些的地理信息系统（GIS）、地图制图学（Cartography）方面的背景知识。由于CartoCSS最终会被解译为mapnik样式并进行制图渲染，所以如果知道并了解mapnik的工作原理，那么会对CartoCSS的语法要素及工作机理有更深刻的理解，但这不是使用CartoCSS进行制图所必需的。关于背景知识，可以参考mapnik项目中的相关文档。

我会尽力保证这份文档与官方英文文档同步。这个仓库中保存了中文版文档的markdown格式版本，以及通过Ulysses生成的pdf与html版本。html的在线版本可以直接从这里查看。

关于翻译的更多信息请参考这篇博客，有问题请与Lu Liu(nudtlliu#gmail.com, luliu.me)联系。

概述

CartoCSS是一种语法类似CSS（Cascading Style Sheets，层叠样式表，一种对网页进行设计的样式语言）的制图样式描述语言。如果熟悉CSS的话，那么CartoCSS这种对地图进行样式设计的语言也会看起来不陌生，尽管二者所包含的要素、属性等内容和含义完全不同。

符号

CartoCSS所基于的地图渲染引擎Mapnik提供了一组基本样式，基于这些基本样式可以构造出复杂的地图样式。这些基本样式被称为**符号**，每种符号都有一系列可配置的属性。

Mapnik中目前包含10种符号。每一种符号都可以应用于以下某一类或几类几何要素：

1. 线符号（可用于线要素和面要素）
2. 面符号（可用于面要素）
3. 点符号（可用于点要素）
4. 文本符号（可用于点要素、线要素和面要素）
5. 盾标符号（可用于点要素和线要素）
6. 线模式（可用于线要素和面要素）
7. 面模式（可用于面要素）
8. 栅格符号（可用于栅格）
9. 注记符号（可用于点要素、线要素和面要素）
10. 建筑物符号

需要注意的是，尽管面符号可以用于定制线要素的样式，但往往会出现不可预期的不理想结果，因此不推荐使用。

Multiple symbolizers can be applied to the same layer - some common combinations are line & polygons, point & text, line & markers, and line & line pattern.

对同一个图层可以同时应用多种符号来定制样式。这种用法我们称之为“多符号”。常用的多符号组合包括：线符号加面符号，点符号、文本符号、线符号加注记符号，以及线符号加线模式等。

A symbolizer is not present on the map unless it has a style defined, but once one of its style properties is added to the stylesheet default values will apply to the other properties for that symbolizer unless overridden. For example, the default line symbolizer color is black, so if you assign a line-width to a layer that line will be black unless you also assign a different color.

一种符号只有在明确定义了它的样式之后才能被绘制在地图上。在每种符号的诸多属性中，除了显式赋值的属性以外，其它属性将全部被设置为默认值。例如，线符号中颜色属性的默认值为黑色，所以如果用户显式设置了线宽，那么图层中的线要素就将以用户设置的宽度被绘制成黑色。

多符号

A single layer is not limited to one of each symbolizer type. For example, multiple semi-transparent line symbolizers can be assigned to a polygon to achieve a soft glow or shadow effect. Multiple text symbolizers can be assigned to the same point with different offsets to label it with more than one field.

对一个图层来说，它的样式可以不局限于仅使用单一的某种符号来定制。举例来说，为了在多边形的边界上获得一种柔和的光晕或阴影效果，可以定义多个半透明线符号，共同发挥作用达到渲染效果。再举一例，对点要素，可以通过定义多个文本符号将若干个属性字段以不同偏移的形式标注在点要素的周围。

Normally when you assign a style to a layer, the style applies to a default symbolizer that is created. In the following example, the second rule overrides the first one because they both apply to the default symbolizer.

通常，如果对一个图层定义了一种样式，那么这种样式就会应用于一种默认的符号。在下面的例子中，后一个样式规则就会将前一个覆盖，因为二者都应用了相同的默认符号，即线符号。

```
#layer {
  line-color: #C00;
  line-width: 1;
}

#layer {
  line-color: #0AF;
  line-opacity: 0.5;
  line-width: 2;
}
```

You can explicitly declare any number of new symbolizers for a layer that will be rendered in addition to styles they would otherwise conflict with. New symbolizers are defined using a double colon syntax inspired by pseudo-elements in CSS3:

用户可以通过显式声明的方式为一个图层增加任意数量的**新符号**。由这些新符号所定义的样式之间只要不互相冲突，那么它们都将被用于渲染该图层。为图层定义新符号使用双冒

号“::”语法，与CSS3中的伪元素定义类似：

```
#layer {
  /* styles for the default symbolizers */
}

#layer::newsymbol {
  /* styles for a new symbolizer named 'newsymbol' */
}
```

Note that newsymbol is not a special keyword but an arbitrary name chosen by the user. To help keep track of different symbolizers you can name additional symbolizers whatever makes sense for the situation. Some examples: `#road::casing`, `#coastline::glow_inner`, `#building::shadow`.

注意上面例子中的newsymbol不是关键字。用户可以为新符号自定义名字，但是为了便于理解，最好取一些有意义的名字，例如：`#road::casing`，`#coastline::glow_inner`，`#building::shadow`。

Returning to our previous example, declaring the second rule will add a blue glow on top of the red line instead of replacing it:

在上一个例子中，我们可以通过再声明一个新符号来实现一个蓝色光晕效果。而正是通过增加了这个新符号的声明，使得蓝色光晕能够被叠加渲染在之前的红色轮廓线之上，而不是覆盖了前面红色线样式（如图）。

```
#layer {
  line-color: #C00;
  line-width: 1;
}

#layer::glow {
  line-color: #0AF;
  line-opacity: 0.5;
  line-width: 4;
}
```

Symbolizers are rendered in the order they are defined, so here the `::glow` (blue line) appears on top of the first style (red line).

在对所定义的符号进行渲染的时候，是按照其在样式脚本中出现的顺序进行的。所以上面例子中的新符号`::glow`（蓝色光晕线）会被绘制在之前定义的红色轮廓线之上。

Named symbolizer styles can still be overridden by further styles that reference the same symbolizer name. In this example, the line color will be green, not green-on-yellow.

具名符号样式也同样会有同名覆盖问题，即后定义的具名符号会覆盖之前先定义的同名具名符号的样式设置。在下面的例子中，线的颜色最终将被渲染为绿色（RGB值为#3F6），而不是半透明黄色上叠加一层绿色效果（如图）。

```
.border::highlight {
  line-color: #FF0;
  line-opacity: 0.5;
}

.border::highlight {
  line-color: #3F6;
}
```

快速入门

TODO: 根据MapBox Studio文档中的QuickStart部分撰写

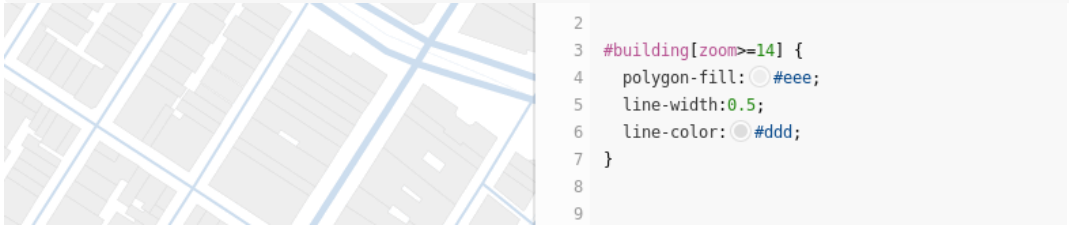
Mapbox Studio uses a language called CartoCSS to determine the look of a map. Colors, sizes, and shapes can all be manipulated by applying their specific CartoCSS parameters in the stylesheet panel to the right of the map. Read the CartoCSS manual for a more detailed introduction to the language.

In this tutorial we'll create a custom style by writing CartoCSS for buildings, roads, and parks.

Styling buildings

Add the following CartoCSS to your custom stylesheet and then click Save.

```
#building[zoom>=14] {
 polygon-fill:#eee;
 line-width:0.5;
 line-color:#ddd;
}
```

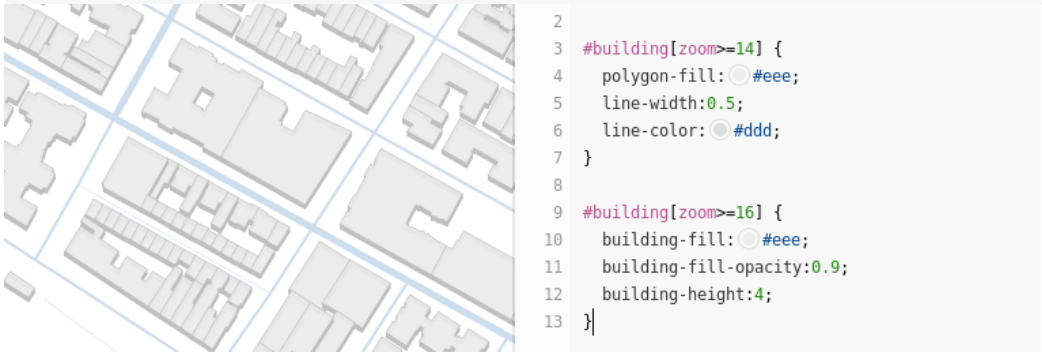


图片来源：www.mapbox.com

- `#building` selects the building layer as the one that will be styled.
- `[zoom>=14]` restricts the styles to zoom level 14 or greater.
- `polygon-fill: #eee` fills the building polygons with a light grey color.

To add depth to our buildings at higher zoom levels let's add another set of rules that use the building symbolizer to render polygons as block-like shapes. Add the following CartoCSS to your custom stylesheet and then click Save.

```
#building[zoom>=16] {
 building-fill:#eee;
 building-fill-opacity:0.9;
 building-height:4;
}
```



图片来源：www.mapbox.com

Styling parks

Add the following CartoCSS to your custom stylesheet and then click Save.

```
#landuse[class='park'] {
 polygon-fill:#dec;
}

#poi_label[maki='park'][scalerank<=3][zoom>=15] {
 text-name:@name;
 text-face-name:@sans;
 text-size:10;
 text-wrap-width: 60;
 text-fill:#686;
 text-halo-fill:#fff;
 text-halo-radius:1;
 text-halo-rasterizer:fast;
}
```


```
14
15 #landuse[class='park'] {
16   polygon-fill: #dec;
17 }
18
19 #poi_label[maki='park'][scalerank<=3][zoom>=15] {
20   text-name:@name;
21   text-face-name:@sans;
22   text-size:10;
23   text-wrap-width: 60;
24   text-fill: #686;
25 }
```

图片来源：www.mapbox.com

- `#landuse` selects features from the landuse layer.
- `[class='park']` restricts the landuse layer to features where the class attribute is park.
- `#poi_label` selects the `poi_label` layer for labelling parks.
- `[maki='park'][scalerank<=3][zoom>=15]` restricts the `poi_label` layer to prominent park labels and restricts their visibility to zoom level 15 or greater.
- `text-name: @name` sets the field that label contents will use for their text. It references the existing `@name` variable defined in the `style` tab.
- `text-face-name: @sans` sets the font to use for displaying labels. It references the existing `@sans` variable defined in the style tab.
- `text-wrap-width: 60` sets a maximum width for a single line of text.
- `text-halo-rasterizer: fast` uses an alternative optimized algorithm for drawing halos around text that improves rendering speed.

Labelling roads

Add the following CartoCSS to your custom stylesheet and then click Save.

```
#road_label[zoom>=13] {
 text-name:@name;
 text-face-name:@sans;
 text-size:10;
 text-placement:line;
 text-avoid-edges:true;
 text-fill:#68a;
 text-halo-fill:#fff;
 text-halo-radius:1;
 text-halo-rasterizer:fast;
}
```


```
26
27 #road_label[zoom>=13] {
28   text-name:@name;
29   text-face-name:@sans;
30   text-size:10;
31   text-placement:line;
32   text-avoid-edges:true;
33   text-fill: #68a;
34   text-halo-fill: #fff;
35   text-halo-radius:1;
36   text-halo-rasterizer:fast;
37 }
38
```

图片来源：www.mapbox.com

- `#road_label` selects features from the road_label layer.
- `[zoom>=13]` restricts the road_label layer to zoom level 13 or greater.
- `text-placement: line` sets labels to follow the orientation of lines rather than horizontally.
- `text-avoid-edges: true` forces labels to be placed away from tile edges to avoid being clipped.

基础用法

样式选择器（Selectors）

CartoCSS styles are constructed by applying blocks of style rules to groups of objects. Style blocks are bounded by curly braces {} and contain style properties and values. Selectors are what allow you restrict these styles to specific layers or groups of objects within layers.

By layer ID

Select all of the objects from a single layer by the layer's ID. Separate multiple layer IDs with commas to select them for a single style.

```
#layer_name {
 // styles
}
#layer_1,
#layer_2 {
 // styles will apply to all the objects in both layers
}
```

By layer class

You can also assign classes to layers to select multiple layers more simply. In Mapbox Studio (unlike TileMill) layer classes are only available for advanced usage.

```
.roads {
 // styles will apply to all layers
 // with a class of 'roads'
}
```

Filter selectors

You can modify selections with filters that reduce the number of objects a style applies to based on certain criteria. Filters let your style read into the various text and numeric properties attached to each object in a layer. For example, you might have all your roads in a single layer, but you could use filters to specify different line colors for different road classifications.

Filters should be written inside square brackets after a layer selector or nested inside a larger style block.

Zoom level filters

Restrict styles to certain zoom levels. This style will only apply when your map is zoomed all the way out to zoom level 0:

```
#layer[zoom=0] { /* style */ }
```

You can specify ranges of zoom levels using two filters:

```
#layer[zoom>=4][zoom<=10] { /* style */ }
```

Valid operators for zoom filters are = (equal to), > (greater than), < (less than), >= (greater than or equal to), <= (less than or equal to), != (not equal to).

You can nest filters to better organize your styles. For example, this style will draw red lines from zoom levels 4 through 10, but the lines will be thicker for zoom levels 8, 9, and 10.

```
#layer[zoom>=4][zoom<=10] {
 line-color: red;
 line-width: 2;
 [zoom=8] { line-width: 3; }
 [zoom=9] { line-width: 4; }
 [zoom=10] { line-width: 5; }
}
```

Numeric value comparison filters

The same comparison operators available for the zoom filter can also be used for any numeric column in your data. For example, you might have a population field in a source full of city points. You could create a style that only labels cities with a population of more than 1 million.

```
#cities[population>1000000] {
 text-name: [name];
 text-face-name: 'Open Sans Regular';
}
```

You could also combine multiple numeric filters with zoom level filters to gradually bring in more populated cities as you zoom in.

```
#cities {
 [zoom>=4][population>1000000],
 [zoom>=5][population>500000],
 [zoom>=6][population>100000] {
    text-name: [name];
    text-face-name: 'Open Sans Regular';
 }
}
```

As with zoom levels, you can select data based on numeric ranges.

```
#cities[population>100000][population<2000000] { /* styles */ }
```

Text comparison filters

You can also filter on columns that contain text. Filter on exact matches with the equals operator (=) or get the inverse results with the not-equal operator (!=). Unlike zoom and numeric values, text values must be quoted with either double or single quotes.

As an example, look at the roads layer in Mapbox Streets (the default vector tile source in Mapbox Studio). It contains a field called class, and each value for this field is one of just a few options such as “motorway”, “main”, and “street”. This makes it a good column to filter on for styling.

```
#roads {
 [class='motorway'] {
       line-width: 4;
 }
 [class='main'] {
       line-width: 2;
 }
 [class='street'] {
       line-width: 1;
 }
}
```

To select everything that is not a motorway you could use the != (“not equal”) operator in the filter:

```
#roads[class!='motorway'] { /* style */ }
```

Regular expression filters

Note: This is an advanced feature that may have negative performance implications.

You can match text in filters based on a pattern using the regular expression operator (=~). This filter will match any text starting with ‘motorway’ (ie, both ‘motorway’ and ‘motorway_link’).

```
#roads[class=~'motorway.*'] { /* style */ }
```

The . represents ‘any character’, and the * means ‘any number of occurrences of the preceding expression. So .* used in combination means ‘any number of any characters’.

为线要素配置样式（Styling lines）

Line styles can be applied to both line and polygon layers. The simplest line styles have just a line-width (in pixels) and a line-color making a single solid line. The default values for these properties are 1 and black respectively if they are not specified.

图片来源：Mapbox

```
// country land borders
#admin_0_lines {
 line-width: 0.75;
 line-color: #426;
}
```

Dashed lines

Simple dashed lines can be created with the line-dasharray property. The value of this property is a comma-separated list of pixel widths that will alternatively be applied to dashes and spaces. This style draws a line with 10 pixel dashes separated by 4 pixel spaces:

图片来源：Mapbox

```
#admin_1_line {
  line-width: 0.5;
  line-color: #426;
  line-dasharray: 10, 4;
}
```

You can make your dash patterns as complex as you want, with the limitation that the dasharray values must all be whole numbers.

图片来源：Mapbox

```
#admin_1_line {
  line-width: 0.5;
  line-color: #426;
  line-dasharray: 10, 3, 2, 3;
```

```
}
```

Caps & Joins

With thicker line widths you'll notice long points at sharp angles and odd gaps on small polygons.

图片来源：Mapbox

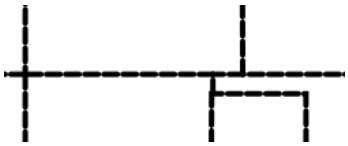
```
#countries::bigoutline {
  line-color: #9ed1dc;
  line-width: 20;
}
```

You can adjust the angles with the `line-join` property: `round` or `square` them off (the default is called `miter`). The gaps can be filled by setting line-cap to `round` or `square` (the default is called `butt`).

图片来源：Mapbox

```
#countries::bigoutline {
  line-color: #9ed1dc;
  line-width: 20;
  line-join: round;
  line-cap: round;
}
```

For dashed lines, line-caps are applied to each dash and their additional length is not included in the dasharray definition. Notice how the following style creates almost-solid lines despite the dasharray defining a gap of 4 pixels.



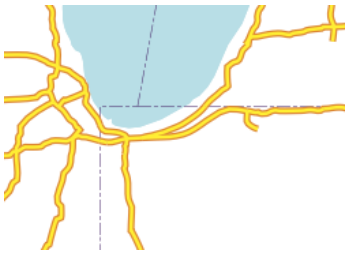
图片来源：Mapbox

```
#layer {
  line-width: 4;
  line-cap: round;
  line-dasharray: 10, 4;
}
```

Compound line styles

Roads

For certain types of line styles you will need to style and overlap multiple line styles. For example, a road with casing:



图片来源：Mapbox

```
#roads[Type='Major Highway'] {
  ::case {
    line-width: 5;
    line-color:#d83;
  }
  ::fill {
    line-width: 2.5;
    line-color:#fe3;
  }
}
```

Dealing with multiple road classes, things get a little more complicated. You can either group your styles by class or group them by attachment. Here we've grouped by class

(filtering on the Type column).

图片来源：Mapbox

```
#roads {
  [Type='Major Highway'] {
    ::case {
      line-width: 5;
      line-color:#d83;
    }
    ::fill {
      line-width: 2.5;
      line-color:#fe3;
    }
  }
  [Type='Secondary Highway'] {
    ::case {
      line-width: 4.5;
      line-color:#ca8;
    }
    ::fill {
      line-width: 2;
      line-color:#ffa;
    }
  }
}
```

Railroads

A common way of symbolizing railroad lines is with regular hatches on a thin line. This can be done with two line attachments - one thin and solid, the other thick and dashed. The dash should be short with wide spacing.

图片来源：Mapbox

```
#railroads {
  ::line, ::hatch { line-color: #777; }
  ::line { line-width:1; }
  ::hatch {
    line-width: 4;
    line-dasharray: 1, 24;
  }
}
```

Another common railroad line style is similar, but with a thin dash and a thick outline. Make sure you define the `::dash` after the `::line` so that it appears on top correctly.

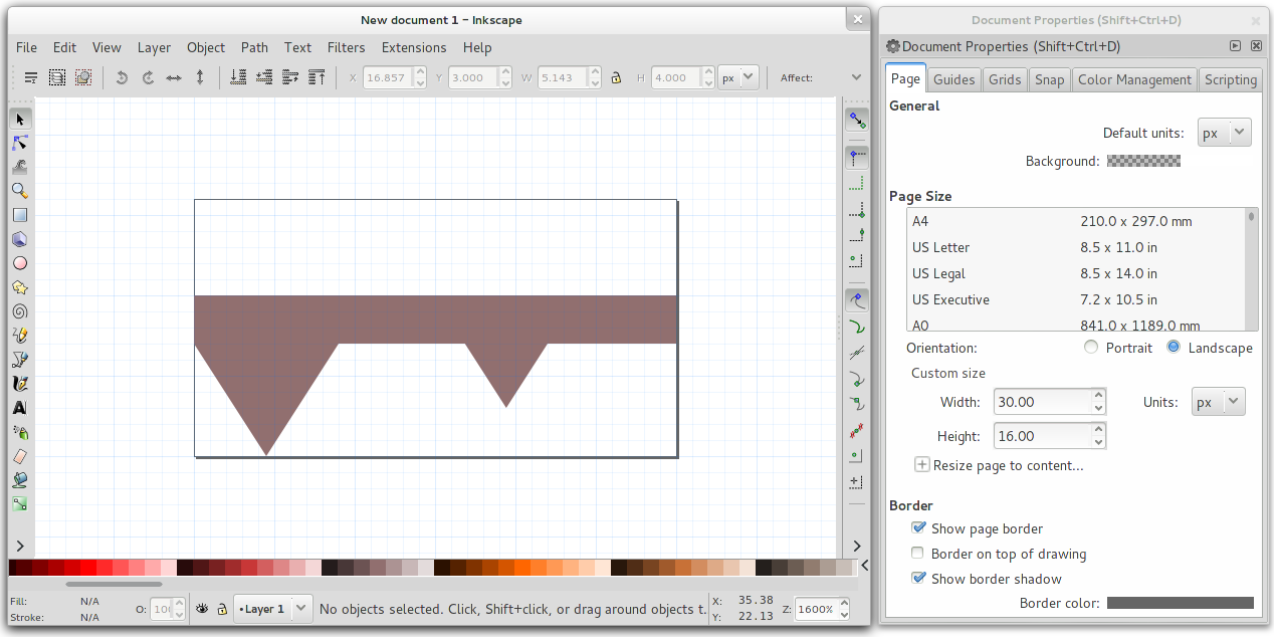
图片来源：Mapbox

```
#railroads {
  ::line {
    line-width: 5;
    line-color: #777;
  }
  ::dash {
    line-color: #fff;
    line-width: 2.5;
    line-dasharray: 6, 6;
  }
}
```

Line patterns with Images

Certain types of line pattens are too complex to be easily achieved with regular compound line styles. TileMill allows you to use repeated images alongside or in place of your other line styles. As an example we'll make a pattern that we'll use to represent a cliff. To do this you'll need to work with external graphics software - we'll be using Inkscape in this example.

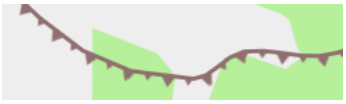
In Inkscape (or whatever you are using), create a new document. The size should be rather small - the height of the image will be the width of the line pattern and the width of the image will be repeated along the length of the line. Our example is 30x16 pixels.



图片来源：Mapbox

Note how the centerline of the pattern is centered on the image (with empty space at the top) for correct alignment with the line data.

To use the image from Inkscape, export it as a PNG file. Line patterns just need a single CartoCSS style to be added to your TileMill project:



图片来源：Mapbox

```
#cliff {
  line-pattern-file: url(cliff.png);
}
```

For some types of patterns, such as the cliff in this example, the direction of the pattern is important. The bottom of line pattern images will be on the right side of lines. The left side of the image will be at the beginning of the line.

为面要素配置样式（Styling polygons）

Polygons are areas that can be filled with a solid color or a pattern, and also given an outline. When you start a new TileMill project with the 'Include world layer and styles' option checked (the default) you'll already have a basic polygon layer and style ready. The fill color of the polygons is defined by the `polygon-fill` property, and it has some line styles as well.

Tip: everything covered in the Styling Lines guide can also be applied to polygon layers.

Basic Styling

If you want to adjust the opacity of a polygon-fill you can use the polygon-opacity property. This is a number between 0 and 1 - 0 being fully transparent and 1 being fully opaque. With 75% opacity we can see both the map background and the lines that have been drawn beneath the polygons.

图片来源：Mapbox

```
#countries {
  polygon-fill: #fff;
  polygon-opacity: 0.75;
}
```

Gaps and Gamma

When you have a layer containing polygons that should fit together seamlessly, you might notice subtle gaps between them at certain scales. You can use the polygon-gamma style to help reduce this effect. Gamma takes a value between 0 and 1 - the default is 1, so try lowering it to hide the gaps. Be careful about setting it too low, though. You'll get jagged edges and possibly even stranger artifacts.

图片来源：Mapbox

```
#countries {
  polygon-fill: #fff;
  polygon-gamma: 0.5;
}
```

Patterns and Textures
With TileMill, you can easily fill areas with textures and patterns by bringing in external images. You might create the patterns yourself in image editing software such as GIMP or Inkscape, or find ready-made images from resource websites such as Subtle Patterns or Free Seamless Textures.

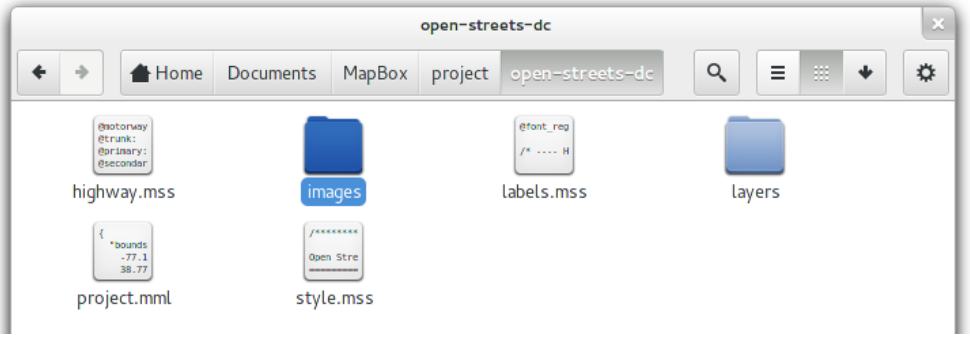
You can add a pattern style from any local file or web URL using the polygon-pattern-file style. Here is a simple diagonal stripe pattern you can use to try out - you can reference it from CartoCSS as in the snippet below.

图片来源：Mapbox

```
polygon-pattern-file: url("http://tilemill.com/assets/pages/pattern-stripe.png");
```

图片来源：Mapbox

For organization it's a good idea to save and store images resources like this on your computer, for example inside your TileMill project folder. To see an example of this look at the Open Streets DC example: open your Documents directory in a file manager and navigate to MapBox→project→open-streets-dc. You can see that there is a subdirectory named 'images' and there are a couple of pattern images inside of it.



图片来源：Mapbox

Images are stored inside the TileMill project they can be *relatively referenced*, meaning you don't need to specify the full path of the file location. Your style would simply be `polygon-pattern-file: url("images/water.png");` . Doing this also makes the TileMill project more portable, for example if you want move it to a different computer.

Global patterns

If you want to add a pattern image to the background of the whole map, you can use the background-image property on the 'Map' object.

```
Map {
 background-image: url("pattern.png");
```

```
}
```

Like all other properties on the Map object, background-image has a global effect - it cannot be filtered or changed depending on zoom level.

If you want to control the a background pattern by zoom level you can add a layer to your project that contains an earth-sized polygon for you to style.

Combining patterns & fills

Using transparency or compositing operations it is possible to get a lot of variety out of a single pattern image.

Ensuring seamlessness

There are two types of pattern alignment: local (the default) and global (specified with `polygon-pattern-alignment: global;`).

When a pattern style has local alignment, that means that the pattern will be aligned separately for each polygon it is applied to. The top-left corner of the pattern image will be aligned to the top-left corner of a polygon's bounding box.

When a pattern style has global alignment, pattern images are aligned to the metatile instead of the geometries. Thus a repeated pattern will line up across all of the polygons it is applied to. With global alignment, pattern images should not be larger than the metatile (excluding the buffer), otherwise portions of the pattern will never be shown.

Another important thing to keep in mind is with globally-aligned patterns is that the pixel dimensions of the image file must multiply evenly up to the width and height of the metatile. If your metatile size is the default of 2, your metatile is 512 pixels wide and tall (2×256). Your pattern width or height dimentions could be 16 or 32 or 128, but should not 20 or 100 or any other number you can't evenly divide 512 by. If you are using patterns from a resource website, you may need to resize them in an image editor to conform to this limitation.

为文本标注配置样式（Styling labels）

原文已更新至新的Mapbox文档体系中：

https://www.mapbox.com/mapbox-studio/styling-labels/

以下还是从原TileMill的文档中摘取过来的。

Basic Point Labels

In CartoCSS, labelling is handled by a variety of properties beginning with `text-`. For each text-related style there are two required properties: `text-name`, which specifies what text goes in the labels, and `text-face-name`, which specifies the typeface(s) will be used to draw the label. (You can see which typefaces are available in the font browser - click the 'A' icon on the left button bar.)

The `text-name` property can pull text from your layer's data fields. If your layer contains a column called `NAME`, a simple label style would look like this:

图片来源：Mapbox

```
#cities {
 text-name: [NAME];
 text-face-name: 'Droid Sans Regular';
}
```

The color and size of these labels will be the defaults - black and 10 pixels respectively. These can be adjusted with the text-fill and text-size properties.

图片来源：Mapbox

```
#cities {
  text-name: [NAME];
  text-face-name: 'Droid Sans Regular';
  text-fill: #036;
  text-size: 20;
}
```

To separate your text from the background, it is often useful to add an outline or halo around the text. You can control the color with text-halo-fill and the width of the halo (in pixels) is controlled with `text-halo-radius`. In the example below, we are using the fadeout color function to make the white halo 30% transparent.

图片来源：Mapbox

```
#cities {
  text-name: [NAME];
  text-face-name: 'Droid Sans Regular';
```

```
  text-fill: #036;
  text-size: 20;
  text-halo-fill: fadeout(white, 30%);
  text-halo-radius: 2.5;
}
```

Text Along Lines

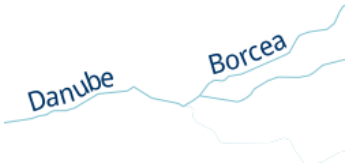
You can also use CartoCSS to style labels that follow a line such as a road or a river. To do this we need to adjust the `text-placement` property. Its default is point; we'll change it to line. We've also added a simple style to visualize the line itself.

图片来源：Mapbox

```
#rivers {
  line-color: #85c5d3;
  text-name: [NAME];
  text-face-name: 'Droid Sans Regular';
  text-fill: #036;
  text-size: 20;
  text-placement: line;
}
```

For rivers it is nice to have the label offset parallel to the line of the river. This can be easily done with the `text-dy` property to specify how large (in pixels) this offset should be. (dy refers to a displacement along the **y** axis.)

We'll also adjust the `text-max-char-angle-delta` property. This allows us to specify the maximum line angle (in degrees) that the text should try to wrap around. The default is 22.5°; setting it lower will make the labels appear along straighter parts of the line.



图片来源：Mapbox

```
#rivers {
  line-color: #85c5d3;
  text-name: [NAME];
  text-face-name: 'Droid Sans Regular';
  text-fill: #036;
  text-size: 20;
  text-placement: line;
  text-dy: 12;
  text-max-char-angle-delta: 15;
}
```

Adding custom text

Labels aren't limited to pulling text from just one column. You can combine data from many columns as well as arbitrary text to construct your `text-name`. For example you could include the state/province separated by a comma and a space.

```
#cities {
  text-name: [NAME] + ', ' + [ADM1NAME];
  text-face-name: 'Droid Sans Regular';
  text-size: 20;
}
```

Other potential uses:

- Multilingual labels: `[name] + '(' + [name_en] + ')'`
- Numeric units: `[elevation] + 'm'`
- Clever unicode icons: `'⚑ ' + [embassy_name] or '⚓ ' + [harbour_name]`

You can also assign arbitrary text to labels that does not come from a data field. Due to a backwards-compatibility issue, you will need to quote such text twice for this to work correctly.

```
#parks {
 text-name: "'Park'";
 text-face-name: 'Droid Sans Regular';
}
```

If you need to include quotation marks in your custom quoted text, you will need to *escape* them with a backslash. For example, for the custom text **City's "Best" Coffee**:

```
text-name: "'City\'s \"Best\" Coffee'";
```

Multi-line labels

You can wrap long labels onto multiple lines with the `text-wrap-width` property which specifies at what pixel width labels should start wrapping. By default the first word that crosses the wrap-width threshold will not wrap - to change this you can set `text-wrap-before` to `true`.

图片来源：Mapbox

```
#cities {
  text-name: [NAME] + ', ' + [ADM1NAME];
  text-face-name: 'Droid Sans Regular';
  text-size: 20;
  text-wrap-width: 100;
  text-wrap-before: true;
}
```

Note that text wrapping not yet supported with `text-placement: line`.

You may have a specific point where you want the line breaks to happen. You can use the text-wrap-character to cause labels to wrap on a character other than a space. With a properly constructed dataset this can give you better control over your labels.

For example we could alter our compound label example to separate the two fields only with an underscore. Setting the wrap character to _ (and also setting a very low wrap width to force wrapping) ensures that the two fields will always be written on their own lines.

图片来源：Mapbox

```
#cities {
  text-name: [NAME] + '_' + [ADM1NAME];
  text-face-name: 'Droid Sans Regular';
  text-size: 20;
  text-wrap-width: 1;
  text-wrap-character: '_';
}
```

Layering Labels

If you are applying label styles to layers that also have line or polygon styles you might notice some unexpected overlapping where the labels aren't necessarily on top.

For simple stylesheets you can control this by making sure your geometry styles and your text styles are in separate attachments:

```
#layer {
  ::shape {
    polygon-fill: #ace;
    line-color: #68a;
  }
  ::label {
    text-name: [name];
    text-face-name: 'Arial Regular';
  }
}
```

However in many cases you'll need to create a label layer that is separate from the layer you use for line and polygon styling. As an example of this, you can look at the Open Streets DC project that comes with TileMill.

The layers roads and `roads-label` reference the same data, but are separated for correct ordering. For more details on how object stacking works in TileMill, see the Symbol Drawing Order guide.

TODO: 这里需要重新组织以消除对TileMill的依赖

关于制图渲染中的各种顺序问题

Objects in TileMill are drawn using a Painter's Algorithm, meaning everything is drawn in a specific order, and things that are drawn first might be covered by things that are drawn later.

Overview

The order in which objects are drawn depends on the following conditions. See the sections that follow for more details.

1. Layers: "Higher" layers obscure "lower" ones.
2. Stylesheets are applied from left to right.
3. Rules within a Stylesheet are applied from top to bottom. This means two things: a). objects matched by later rules will be drawn over those defined by earlier rules; and b). attachments may be redefined by later rules.

4. Attachments (e.g., `::glow { ... }`) within a Stylesheet are applied from top to bottom.
5. Lastly, all else being equal, objects are drawn in the order in which they are found, such as in PostGIS.

Order vs. Priority

For things like lines and areas, objects that are drawn first are less likely to be fully visible. Objects high in the stack might completely obscure other objects, thus you might associate these with a high 'priority' or 'importance'.

However for things like text, markers, and icons that have their allow-overlap properties set to false (the default) things work a bit differently. Objects that are drawn first are **more** likely to be visible; instead of letting things sit on top of each other, overlapping objects are simply skipped. Since such objects higher in the stack are less likely to be drawn, you might associate these with a low 'priority' or 'importance'.

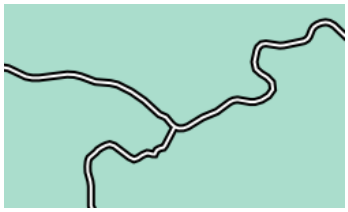
Layer Ordering

Layers are the most basic and explicit way of controlling the order of elements on a map in TileMill. Layers are rendered in order starting at the bottom of the layers list moving up. You can adjust the position of each layer in the stack by clicking and draggin on its geometry icon.

If you look at the example projects that come with TileMill, you can see that the basic parts of the map (eg. landuse areas, water) are in layers at the bottom of the list. The things that shouldn't be covered up by anything else (eg. labels, icons) are in layers at the top of the list.

Attachment Ordering

Within a layer, styles can be broken up into 'attachments' with the `:: syntax`. Think of attachments like sub-layers.

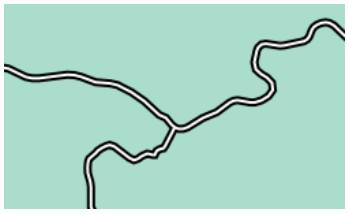


图片来源：Mapbox

```
#layer {
  ::outline {
    line-width: 6;
    line-color: black;
  }
  ::inline {
    line-width: 2;
    line-color: white;
  }
}
```

Attachments are drawn in the order they are first defined, so in the example above the `::outline` lines will be drawn below the `::inline` lines.

Note that all styles are nested inside attachments. If you don't explicitly define one, a default attachment still exists. Thus the following style produces the same result as the one above.



图片来源：Mapbox

```
#layer {
  ::outline {
    line-width: 6;
    line-color: black;
  }
  line-width: 2;
  line-color: white;
}
```

Data Ordering

Within each attachment, the order that your data is stored/retrieved in is also significant.

When styling city labels, for example, it's good to ensure that the order of your data makes sense for label prioritization. For data coming from an SQL database you should ORDER BY a population column or some other prioritization field in the select statement.

Data coming from files are read from the beginning of the file to the end and cannot be re-ordered on-the-fly by TileMill. You'll want to pre-process such files to make sure the ordering makes sense.

You can do this from the terminal with ogr2ogr (see Setting up GDAL). This example rearranges all the objects in cities.shp based on the population field in descending order (highest population first).

```
ogr2ogr -sql \
```

```
  'select * from cities order by population desc' \
  cities_ordered.shp cities.shp
```

Symbolizer Ordering

Each object in each attachment may have multiple symbolizers applied to it. That is, a polygon might have both a fill and an outline. In this case, the styles are drawn in the same order they are defined.

In this style, the outline will be drawn below the fill:

图片来源：Mapbox

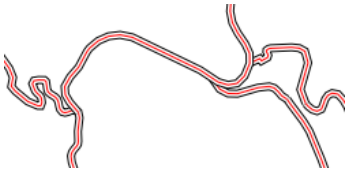
```
#layer {
  line-width: 6;
  polygon-fill: #aec;
  polygon-opacity: 0.8;
}
```

In this style, the line is drawn on top of the fill:

图片来源：Mapbox

```
#layer {
  polygon-fill: #aec;
  polygon-opacity: 0.8;
  line-width: 6;
}
```

It's also possible to create multiple symbols of the same type within an attachment using named *instances*. Like attachments, their names are arbitrary.



图片来源：Mapbox

```
#layer {
  bottomline/line-width: 6;
  middleline/line-width: 4;
  middleline/line-color: white;
  topline/line-color: red;
}
```

Note that symbolizer ordering happens after all other types of ordering - so an outline might be on top of one polygon but beneath a neighboring polygon. If you want to ensure lines are always below fills, use separate attachments.

Layer Source Ordering

The order in which your layer sources are specified also influences rendering order: data from sources are rendered in order. So if you click "Change source" under "Layers" and you see you.id123,mapbox.mapbox-terrain-v1, the layers from mapbox.mapbox-terrain-v1 will render last, over the layers from you.id123. To ensure that your own data renders last, use mapbox.mapbox-terrain-v1,you.id123.

关于合成操作（Compositing）

Compositing operations affect the way colors and textures of different elements and styles interact with each other.

Without any compositing operations on a source it will just be painted directly over the destination – compositing operations allow us to change this. There are 33 compositing operations available in CartoCSS:

Compositing operations in CartoCSS		
plus	difference	src
minus	exclusion	dst
multiply	contrast	src-over
screen	invert	dst-over
overlay	invert-rgb	src-in
darken	grain-merge	dst-in
lighten	grain-extract	src-out
color-dodge	hue	dst-out
color-burn	saturation	src-atop
hard-light	color	dst-atop
soft-light	value	xor

高级技巧

高级地图设计

参考 https://www.mapbox.com/tilemill/docs/guides/advanced-map-design/

高级标注符号用法

参考 https://www.mapbox.com/tilemill/docs/guides/labels-advanced/

CartoCSS语言参考

The following is a list of properties provided in CartoCSS that you can apply to map elements.

CartoCSS提供了一系列用于定义地图样式的属性。以下列表中包含了这些属性的含义和所有可取的值。

所有符号的公共属性

image-filters `functions`

默认值： none
(不使用图像过滤器)

A list of image filters that will be applied to the active rendering canvas for a given style. The presence of one more more `image-filters` will trigger a new canvas to be created before starting to render a style and then this canvas will be composited back into the main canvas after rendering all features and after all `image-filters` have been applied. See `direct-image-filters` if you want to apply a filter directly to the main canvas.

以函数形式提供的一组图像过滤器。图像过滤器会作用于处于活动状态的画布。如果设置了多个图像过滤器，那么每增加一个过滤器都会触发创建一个新的画布，当这个新的画布被渲染完成后，再通过合成的方式与主画布合并。如果要直接在主画布上应用图像过滤器，那么需要使用 `direct-image-filters` 属性。

---

direct-image-filters `functions`

默认值： none
(不使用图像过滤器)

A list of image filters to apply to the main canvas (see the `image-filters` doc for how they work on a separate canvas)

作用于主画布上的图像过滤器（参见 `image-filters` ）

---

comp-op `keyword`

`clear``src``dst``src-over``dst-over``src-in``dst-in``src-out``dst-out``src-atop``dst-atop``xor``plus``minus``multiply``screen``overlay``darken``lighten``color-dodge``color-burn``hard-light``soft-light``difference``exclusion``contrast``invert``invert-rgb``grain-merge``grain-extract``hue``saturation``color``value`

默认值： src-over
(把当前图层覆盖在其它图层之上)

Composite operation. This defines how this layer should behave relative to layers atop or below it.

合成操作。该属性用于定义当前图层与其相邻图层如何合成。关于合成运算，请参见图像合成（Image Compositing）技术的相关内容。

---

opacity `float`

默认值： 1
(no separate buffer will be used and no alpha will be applied to the style after rendering)

不使用单独缓冲区，不设置alpha通道

An alpha value for the style (which means an alpha applied to all features in separate buffer and then composited back to main buffer)

为样式设置alpha值（首先，创建一个独立的缓冲区，在这个缓冲区中为所有要素应用alpha实现透明化，然后再把这个独立缓冲区合成到主缓冲区中）

地图（map）的属性

background-color `color`

默认值： none
(透明色)

Map Background color

设置地图的背景颜色

---

background-image `uri`

默认值：
(透明色)

An image that is repeated below all features on a map as a background.

一张以平铺形式置于最底层的背景图片。

---

background-image-comp-op `keyword`

`clear``src``dst``src-over``dst-over``src-in``dst-in``src-out``dst-out``src-atop``dst-atop``xor``plus``minus``multiply``screen``overlay``darken``lighten``color-dodge``color-burn``hard-light``soft-light``difference``exclusion``contrast``invert``invert-rgb``grain-merge``grain-extract``hue``saturation``color``value`

默认值： src-over
(The background-image will be blended with the background normally (placed on top of any existing background-color))

背景图片被置于所设置的背景色上一层。

Set the compositing operation used to blend the image into the background

设置背景图片与背景颜色之间的合成操作方式。

---

background-image-opacity `float`

默认值： 1
(The image opacity will not be changed when applied to the map background)

背景图片的透明度保持不变。

Set the opacity of the image

设置图片的透明度

---

srs `string`

默认值： +proj=longlat +ellps=WGS84 +datum=WGS84 +no_defs
(The proj4 literal of EPSG:4326 is assumed to be the Map's spatial reference and all data from layers within this map will be plotted using this coordinate system. If any layers do not declare an srs value then they will be assumed to be in the same srs as the Map and not transformations will be needed to plot them in the Map's coordinate space)

这是EPSG:4326空间参考系的proj4表达形式。地图中所有图层的数据都会采用这同一种参考系来绘制。如果地图中的某一图层没有显式声明自己所使用的参考系，那么这个图层将被认为与地图的参考系相同，并且在绘制的时候不会对这个图层中包含的数据进行座标变换。

Map spatial reference (proj4 string)

地图的空间参考系（以proj4字符串表达）

---

buffer-size `float`

默认值： 0
(无缓冲区)

Extra tolerance around the map (in pixels) used to ensure labels crossing tile boundaries are equally rendered in each tile (e.g. cut in each tile). Not intended to be used in combination with "avoid-edges".

在地图周围增加一圈额外的绘制区域（以像素数表达）。这个属性的设置是为了保证那些出现在地图边界附近的文本标注不至于在渲染时被截断。注意这个属性不应该与“avoid-edges”同时使用。

---

base `string`

默认值:
(This base path defaults to an empty string meaning that any relative paths to files referenced in styles or layers will be interpreted relative to the application process.)

工作路径默认为空。此时在样式定义中所有通过相对路径的方式引用的外部资源文件都会以应用程序所在的路径为父目录去寻址。

Any relative paths used to reference files will be understood as relative to this directory path if the map is loaded from an in memory object rather than from the filesystem. If the map is loaded from the filesystem and this option is not provided it will be set to the directory of the stylesheet.

如果map是从内存中加载的（？），那么所有以相对路径方式引用的外部资源则均为相对于由该base属性定义的路径。如果map是从文件系统中加载的，并且这个base属性没有被显式设置，那么base的值就是样式文件所在的目录。

---

font-directory `uri`

默认值： none
(No map-specific fonts will be registered)

不注册专门用于当前map的字体

Path to a directory which holds fonts which should be registered when the Map is loaded (in addition to any fonts that may be automatically registered).

指定专门用于当前map的字体目录（自动加载的默认字体除外）

线符号（line）的属性

line-color `color`

默认值： rgba(0,0,0,1)
(black and fully opaque (alpha = 1), same as rgb(0,0,0))

完全不透明的黑色

The color of a drawn line

设置线要素的线条颜色。

---

line-width `float`

默认值： 1

The width of a line in pixels

设置线要素的线条宽度，单位为像素。

---

line-opacity `float`

默认值： 1
(opaque)

不透明

The opacity of a line

设置线要素的透明度。

---

line-join `keyword`

取值范围： `miter``round``bevel`

默认值： miter

The behavior of lines when joining

设置线要素之间在交汇点处如何绘制。三种绘制方法的比较如下图所示。

（图片来源：www.w3.org）

---

line-cap `keyword`

`butt``round``square`

默认值： butt

The display of line endings

设置线要素的端点形状。
* * *

line-gamma `float`

默认值： 1
(fully antialiased)

完全抗锯齿

Range: 0-1

取值范围：0到1

Level of antialiasing of stroke line

设置绘制线要素时的抗锯齿级别。

---

line-gamma-method `keyword`

`power``linear``none``threshold``multiply`

默认值： power
(pow(x,gamma) is used to calculate pixel gamma, which produces slightly smoother line and polygon antialiasing than the 'linear' method, while other methods are usually only used to disable AA)

使用pow(x, gamma)来计算像素gamma值。与linear相比，应用power值绘制出来的线与面要素更加平滑。而其它的取值通常只是用来关闭抗锯齿。

An Antigrain Geometry specific rendering hint to control the quality of antialiasing. Under the hood in Mapnik this method is used in combination with the 'gamma' value (which defaults to 1). The methods are in the AGG source at https://github.com/mapnik/mapnik/blob/master/deps/agg/include/agg_gamma_functions.h

设置抗锯齿的具体算法，控制绘制质量。

---

line-dasharray `numbers`

默认值： none
(solid line)

实线（无虚线效果）

A pair of length values [a,b], where (a) is the dash length and (b) is the gap length respectively. More than two values are supported for more complex patterns.

通过设置[a,b]的值设置虚线样式。其中a为虚线段的长度，b为虚线段间隔的长度。此外，还可以设置更多的值，从而获得更加复杂的绘制效果。

---

line-miterlimit `float`

默认值： 4
(Will auto-convert miters to bevel line joins when theta is less than 29 degrees as per the SVG spec: 'miterLength / stroke-width = 1 / sin (theta / 2)')

当theta角小于29度时，自动将线要素交汇样式从miter改为bevel

The limit on the ratio of the miter length to the stroke-width. Used to automatically convert miter joins to bevel joins for sharp angles to avoid the miter extending beyond the thickness of the stroking path. Normally will not need to be set, but a larger value can sometimes help avoid jaggy artifacts.

设置线端的切角长度与线宽的比例上限。当线要素交汇处出现尖锐的锐角时，由于切角与线宽比例失调会导致错误的绘制结果。设置了该属性则会在出现上述情况时自动将线要素交汇样式从miter改为bevel。一般情况下，这个属性不需要显式设置，但有时可以通过设定一个较大的值可以避免出现参差不齐的不良绘制效果。

---

line-clip `boolean`

默认值： true
(geometry will be clipped to map bounds before rendering)

几何要素会根据地图的地理范围进行切割。

geometries are clipped to map bounds by default for best rendering performance. In some cases users may wish to disable this to avoid rendering artifacts.

为了提高绘制效率，可以先将矢量要素中所有超出地图边界的部分切掉，再进行绘制。但在某些情况下，为了防止出现绘制错误，也可以通过将该值设为false而不采用这个策略。

---

line-simplify `float`

默认值： 0
(geometry will not be simplified)

不对几何要素进行简化

geometries are simplified by the given tolerance

如果要对几何要素按照地图综合的方法进行简化，那么通过该属性来设定阈值。

line-simplify-algorithm `keyword`

`radial-distance``zhao-saalfeld``visvalingam-whyatt`

默认值： radial-distance
(geometry will not be simplified using the radial distance algorithm)

不使用radial-distance算法进行简化（？）

geometries are simplified by the given algorithm

设置对线要素进行综合的简化算法。

---

line-smooth `float`

默认值： 0
(no smoothing)

不对拐点进行平滑

Range: 0-1

取值范围：0到1

Smooths out geometry angles. 0 is no smoothing, 1 is fully smoothed. Values greater than 1 will produce wild, looping geometries.

对线的拐点进行平滑处理。0表示不进行平滑，1表示完全平滑。如果取值大于1，会导致绘制的几何要素扭曲变形。

---

line-offset `float`

默认值： 0
(no offset)

无偏移

Offsets a line a number of pixels parallel to its actual path. Positive values move the line left, negative values move it right (relative to the directionality of the line).

将线要素相对于其原有位置向左（沿着线的走向）或向右偏移一定量的像素绘制。正值表示左偏，负值表示右偏。

---

line-rasterizer `keyword`

`full``fast`

默认值： full

Exposes an alternate AGG rendering method that sacrifices some accuracy for speed.

设置线渲染方式，可以通过牺牲部分精确度以换取绘制速度。

---

line-geometry-transform `functions`

默认值： none
(geometry will not be transformed)

不对几何要素进行变换

Allows transformation functions to be applied to the geometry.

为几何要素定义变换函数。

---

line-comp-op `keyword`

`clear``src``dst``src-over``dst-over``src-in``dst-in``src-out``dst-out``src-atop``dst-atop``xor``plus``minus``multiply``screen``overlay``darken``lighten``color-dodge``color-burn``hard-light``soft-light``difference``exclusion``contrast``invert``invert-rgb``grain-merge``grain-extract``hue``saturation``color``value`

默认值： src-over
(add the current symbolizer on top of other symbolizer)

将当前符号置于其它符号的上一层

Composite operation. This defines how this symbolizer should behave relative to symbolizers atop or below it.

这也是一个合成操作。它定义了当前的符号应该如何与其相邻图层进行合成。
* * *

面符号（polygon）的属性

polygon-fill `color`

默认值： rgba(128,128,128,1)
(gray and fully opaque (alpha = 1), same as rgb(128,128,128))

完全不透明的灰色

Fill color to assign to a polygon

设置面要素的填充色

---

polygon-opacity `float`

默认值： 1
(opaque)

不透明

The opacity of the polygon

面要素的透明度（0为完全透明，1为完全不透明）

---

polygon-gamma `float`

默认值： 1
(fully antialiased)

完全抗锯齿

Range: 0-1

取值范围：0到1

Level of antialiasing of polygon edges

设置面要素边缘的抗锯齿级别，影响绘制效果和速度。抗锯齿级别越高（最高为1），绘制效果越好，但绘制速度最慢；反之，绘制效果最差，但绘制速度最快。注意这里所说的绘制速度的快慢只是理论上的，实际效果与软硬件环境密切相关。

---

polygon-gamma-method `keyword`

`power``linear``none``threshold``multiply`

默认值： power
(pow(x,gamma) is used to calculate pixel gamma, which produces slightly smoother line and polygon antialiasing than the 'linear' method, while other methods are usually only used to disable AA)

使用pow(x, gamma)来计算像素gamma值。与linear相比，应用power值绘制出来的线与面要素更加平滑。而其它的取值通常只是用来关闭抗锯齿。

An Antigrain Geometry specific rendering hint to control the quality of antialiasing. Under the hood in Mapnik this method is used in combination with the 'gamma' value (which defaults to 1). The methods are in the AGG source at https://github.com/mapnik/mapnik/blob/master/deps/agg/include/agg_gamma_functions.h

设置抗锯齿的具体算法，控制绘制质量。

---

polygon-clip `boolean`

默认值： true
(geometry will be clipped to map bounds before rendering)

几何要素会根据地图的地理范围进行切割。

geometries are clipped to map bounds by default for best rendering performance. In some cases users may wish to disable this to avoid rendering artifacts.

为了提高绘制效率，可以先将矢量要素中所有超出地图边界的部分切掉，再进行绘制。但在某些情况下，为了防止出现绘制错误，也可以通过将该值设为false而不采用这个策略。

---

polygon-simplify `float`

默认值： 0
(geometry will not be simplified)

不对面要素的边线进行简化

geometries are simplified by the given tolerance

如果要对面要素的边线按照地图综合的方法进行简化，那么通过该属性来设定阈值。（参见地图综合中的线简化算法，如Douglas-Peuker算法）

---

polygon-simplify-algorithm `keyword`

`radial-distance``zhao-saalfeld``visvalingam-whyatt`

默认值： radial-distance
(geometry will not be simplified using the radial distance algorithm)

不使用radial-distance算法进行简化（？为什么是不简化？）

geometries are simplified by the given algorithm

设置对面要素的边线进行综合的简化算法。

---

polygon-smooth `float`

默认值： 0
(no smoothing)

不对拐点进行平滑

Range: 0-1

取值范围：0到1

Smooths out geometry angles. 0 is no smoothing, 1 is fully smoothed. Values greater than 1 will produce wild, looping geometries.

对线的拐点进行平滑处理。0表示不进行平滑，1表示完全平滑。如果取值大于1，会导致绘制的几何要素扭曲变形。

---

polygon-geometry-transform `functions`

默认值： none
(geometry will not be transformed)

不对几何要素进行变换。

Allows transformation functions to be applied to the geometry.

为几何要素定义变换函数。

---

polygon-comp-op `keyword`

`clear``src``dst``src-over``dst-over``src-in``dst-in``src-out``dst-out``src-atop``dst-atop``xor``plus``minus``multiply``screen``overlay``darken``lighten``color-dodge``color-burn``hard-light``soft-light``difference``exclusion``contrast``invert``invert-rgb``grain-merge``grain-extract``hue``saturation``color``value`

默认值： src-over
(add the current symbolizer on top of other symbolizer)

将当前符号置于其它符号的上一层

Composite operation. This defines how this symbolizer should behave relative to symbolizers atop or below it.

这也是一个合成操作。它定义了当前的符号应该如何与其相邻图层进行合成。

点符号（point）的属性

point-file `uri`

默认值： none

Image file to represent a point
* * *

point-allow-overlap `boolean`

默认值： false
(Do not allow points to overlap with each other - overlapping markers will not be shown.)

Control whether overlapping points are shown or hidden.
* * *

point-ignore-placement `boolean`

默认值： false
(do not store the bbox of this geometry in the collision detector cache)

value to control whether the placement of the feature will prevent the placement of other features
* * *

point-opacity `float`

默认值： 1
(Fully opaque)

A value from 0 to 1 to control the opacity of the point
* * *

point-placement `keyword`

`centroid``interior`

默认值： centroid

How this point should be placed. Centroid calculates the geometric center of a polygon, which can be outside of it, while interior always places inside of a polygon.
* * *

point-transform `functions`

默认值：
(No transformation)

SVG transformation definition
* * *

point-comp-op `keyword`

`clear``src``dst``src-over``dst-over``src-in``dst-in``src-out``dst-out``src-atop``dst-atop``xor``plus``minus``multiply``screen``overlay``darken``lighten``color-dodge``color-burn``hard-light``soft-light``difference``exclusion``contrast``invert``invert-rgb``grain-merge``grain-extract``hue``saturation``color``value`

默认值： src-over
(add the current symbolizer on top of other symbolizer)

Composite operation. This defines how this symbolizer should behave relative to symbolizers atop or below it.
* * *

文本符号（text）的属性

text-name `expression`

默认值：

Value to use for a text label. Data columns are specified using brackets like column_name
设置文本符号上显示的文字。可以通过用中括号括起来的字段名来指定要使用的数据字段，例如[column_name]。
* * *

text-face-name `string`

默认值： undefined

Font name and style to render a label in
设置文本符号所使用的字体。
* * *

text-size `float`

默认值： 10

Text size in pixels
设置文本符号中文字的字号，以像素为单位。
* * *

text-ratio `unsigned`

默认值： 0

Define the amount of text (of the total) present on successive lines when wrapping occurs
设置折行后的文本所占比例（译注：这解释的好扭曲，到底什么意思，有什么意义？）
* * *

text-wrap-width `unsigned`

默认值： 0

Length of a chunk of text in characters before wrapping text
设置文本块在多长的时候进行折行，以字符为单位
* * *

text-wrap-before `boolean`

默认值： false

Wrap text before wrap-width is reached. If false, wrapped lines will be a bit longer than wrap-width.
控制文本文字的折行动作。如果该值为false，那么每一行文本都会比wrap-width属性设定的值略长。
* * *

text-wrap-character `string`

默认值：

Use this character instead of a space to wrap long text.
设置盾标文本的折行字符。
* * *

text-spacing `unsigned`

默认值： undefined

Distance between repeated text labels on a line (aka. label-spacing)
设置沿线绘制文本符号时每两个文本符号之间的间距。
* * *

text-character-spacing `float`

默认值： 0

Horizontal spacing adjustment between characters in pixels
设置文本文字的字间距，以像素为单位。
* * *

text-line-spacing `unsigned`

默认值： 0

Vertical spacing adjustment between lines in pixels
设置文本文字的行间距。
* * *

text-label-position-tolerance `unsigned`

默认值： 0

Allows the label to be displaced from its ideal position by a number of pixels (only works with placement:line)
设置文本标注相对于其理想位置的偏移量，以像素为单位（目前仅适用于线要素）
* * *

text-max-char-angle-delta `float`

默认值： 22.5

The maximum angle change, in degrees, allowed between adjacent characters in a label. This value internally is converted to radians to the default is 22.5*math.pi/180.0. The higher the value the fewer labels will be placed around around sharp corners.
设置文本文字的最大折转角，以十进制角度为单位。这个值会在绘制时被换算成弧度，例如默认的22.5度会按照22.5/math.pi*180.0公式被换算成0.3925弧度。这个值越大，则被绘制在尖锐转角处的文本符号会越少。
* * *

text-fill `color`

默认值： #000000
(black)
黑色

Specifies the color for the text
设置文本文字的颜色。
* * *

text-opacity `float`

默认值： 1
(Fully opaque)
不透明

A number from 0 to 1 specifying the opacity for the text
设置文本文字的透明度，取值范围为0到1。
* * *

text-halo-fill `color`

默认值： #FFFFFF
(white)
白色

Specifies the color of the halo around the text.
设置文本文字边缘的光晕颜色。
* * *

text-halo-radius `float`

默认值： 0
(no halo)
无光晕

Specify the radius of the halo in pixels
设置文本文字边缘的光晕大小，以像素为单位。
* * *

text-halo-rasterizer `keyword`

`full``fast`

默认值： full

Exposes an alternate text halo rendering method that sacrifices quality for speed.
设置用于渲染文字光晕的方法，速度优先还是质量优先。

* * *

text-dx `float`

默认值： 0

Displace text by fixed amount, in pixels, +/- along the X axis. A positive value will shift the text right
设置文本文字的水平偏移量，以像素为单位。正值表示向右偏移。

---

text-dy `float`

默认值： 0

Displace text by fixed amount, in pixels, +/- along the Y axis. A positive value will shift the text down
设置文本文字的垂直偏移量，以像素为单位。正值表示向下偏移。
* * *

text-vertical-alignment `keyword`

`top``middle``bottom``auto`

默认值： auto
(Default affected by value of dy; "bottom" for dy>0, "top" for dy<0.)
自动，但受到dy值的影响。当dy>0时，取bottom；而当dy<0时，取top

Position of label relative to point position.
设置文本符号相对于点要素座标的位置。
* * *

text-avoid-edges `boolean`

默认值： false

Avoid placing labels that intersect with tile boundaries.
设置是否避免将文本标注置于绘制区域（通常为瓦片）的边缘处。
* * *

text-min-distance `float`

默认值： undefined

Minimum permitted distance to the next text symbolizer.
设置文本符号之间的最小间距。
* * *

text-min-padding `float`

默认值： undefined

Minimum distance a text label will be placed from the edge of a metatile.
设置文本符号在元瓦片中的最小边距。
* * *

text-min-path-length `float`

默认值： 0
(place labels on all paths)
无论路线长度是多少，都要绘制文本符号

Place labels only on paths longer than this value.
如果设置了该值，那么只有在当路线长度大于该值的时候才绘制文本符号。
* * *

text-allow-overlap `boolean`

默认值： false
(Do not allow text to overlap with other text - overlapping markers will not be shown.)
不允许文本符号相互压盖

Control whether overlapping text is shown or hidden.
设置是否显式相互压盖的文本符号。
* * *

text-orientation `expression`

默认值： undefined

Rotate the text.
设置文本旋转。
* * *

text-placement `keyword`

`point``line``vertex``interior`

默认值： point

Control the style of placement of a point versus the geometry it is attached to.
设置文本符号在对应几何要素上的放置方式。
* * *

text-placement-type `keyword`

`dummy``simple`

默认值： dummy

Re-position and/or re-size text to avoid overlaps. "simple" for basic algorithm (using text-placements string,) "dummy" to turn this feature off.
设置文本符号之间相互避让的算法。simple表示使用由text-placements属性指定的基本算法。而dummy则表示不使用该特性。
* * *

text-placements `string`

默认值：

If "placement-type" is set to "simple", use this "POSITIONS,SIZES" string. An example is `text-placements: &quot;E,NE,SE,W,NW,SW&quot;;`
如果placement-type属性被设置为simple，那么就会依据该属性的值（即形如“POSITIONS, [SIZES]”的字符串）执行文本符号相互避让算法。例如： `text-placements: "E,NE,SE,W,NW,SW";`
* * *

text-transform `keyword`

`none``uppercase``lowercase``capitalize`

默认值： none

Transform the case of the characters
设置是否对文本字符进行大小写转换。
* * *

text-horizontal-alignment `keyword`

`left``middle``right``auto`

默认值： auto

The text's horizontal alignment from its centerpoint
设置文本文字的水平对齐方式。
* * *

text-align `keyword`

`left``right``center``auto`

默认值： auto
(Auto alignment means that text will be centered by default except when using the `placement-type` parameter - in that case either right or left justification will be used automatically depending on where the text could be fit given the `text-placements` directives)
默认的自动方式是居中对齐，但如果已经设置了 `placement-type` 属性，那么就会依据 `text-placements` 属性的值来对文字进行靠左或靠右对齐

Define how text is justified
设置文本文字的对齐方式。
* * *

text-clip `boolean`

默认值： true
(geometry will be clipped to map bounds before rendering)
几何要素会根据地图的地理范围进行切割

geometries are clipped to map bounds by default for best rendering performance. In some cases users may wish to disable this to avoid rendering artifacts.
为了提高绘制效率，可以先将矢量要素中所有超出地图边界的部分切掉，再进行绘制。但在某些情况下，为了防止出现绘制错误，也可以通过将该值设为false而不采用这个策略。
* * *

text-comp-op `keyword`

`clear``src``dst``src-over``dst-over``src-in``dst-in``src-out``dst-out``src-atop``dst-atop``xor``plus``minus``multiply``screen``overlay``darken``lighten``color-dodge``color-burn``hard-light``soft-light``difference``exclusion``contrast``invert``invert-rgb``grain-merge``grain-extract``hue``saturation``color``value`

默认值： src-over
(add the current symbolizer on top of other symbolizer)
将当前符号置于其它符号的上一层

Composite operation. This defines how this symbolizer should behave relative to symbolizers atop or below it.
这也是一个合成操作。它定义了当前的符号应该如何与其相邻图层进行合成。
* * *

盾标符号（shield）的属性

shield-name `expression`

默认值： undefined

Value to use for a shield"s text label. Data columns are specified using brackets like [column_name]

设置盾标上显示的标注文字。可以通过用中括号括起来的字段名来指定要使用的数据字段，例如[column_name]。
* * *

shield-file `uri`

默认值： none

Image file to render behind the shield text

设置显示在盾标文本后面的背景图片。

---

shield-face-name `string`

默认值:

Font name and style to use for the shield text

设置盾标上标注文字的字体与样式。

---

shield-unlock-image `boolean`

默认值： false
(text alignment relative to the shield image uses the center of the image as the anchor for text positioning.)

盾标文字将被置于盾标背景图片的中心位置

This parameter should be set to true if you are trying to position text beside rather than on top of the shield image

设置盾标文字与背景图片之间的位置关系。如果不想把盾标文本绘制在背景图的中心，那么就应该将该属性值设置为true。

---

shield-size `float`

默认值： undefined

The size of the shield text in pixels

设置盾标文字的大小，以像素为单位。
* * *

shield-fill `color`

默认值： undefined

The color of the shield text

设置盾标文字的颜色。
* * *

shield-placement `keyword`

`point``line``vertex``interior`

默认值： point

How this shield should be placed. Point placement attempts to place it on top of points, line places along lines multiple times per feature, vertex places on the vertexes of polygons, and interior attempts to place inside of polygons.

设置盾标的放置方式。point方式是将盾标置于点要素的位置，line方式是将盾标在线要素上沿线绘制多次，vertex方式是将盾标置于多边形的顶点位置，而interior方式则是将盾标置于面要素的内部。

---

shield-avoid-edges `boolean`

默认值： false

Avoid placing shields that intersect with tile boundaries.

设置是否避免在地图或瓦片的边缘处绘制盾标。

---

shield-allow-overlap `boolean`

默认值： false
(Do not allow shields to overlap with other map elements already placed.)

不允许盾标与其它现有地图要素重叠

Control whether overlapping shields are shown or hidden.

该属性用于设置在盾标与地图上其它符号出现压盖时，是否显示盾标。
* * *

shield-min-distance `float`

默认值： 0

Minimum distance to the next shield symbol, not necessarily the same shield.
设置相邻两个盾标符号（可以是相同的盾标，也可以是不同的）之间的最小距离
* * *

shield-spacing `float`

默认值： 0

The spacing between repeated occurrences of the same shield on a line
设置在同一线要素上多次绘制的盾标之间的间隔。
* * *

shield-min-padding `float`

默认值： 0

Minimum distance a shield will be placed from the edge of a metatile.
设置盾标在瓦片上绘制时的最小边距。
* * *

shield-wrap-width `unsigned`

默认值： 0

Length of a chunk of text in characters before wrapping text

设置盾标文本多长的时候需要折行。
* * *

shield-wrap-before `boolean`

默认值： false

Wrap text before wrap-width is reached. If false, wrapped lines will be a bit longer than wrap-width.

控制盾标文本的折行动作。如果该值为false，那么每一行文本都会比wrap-width属性设定的值略长。
* * *

shield-wrap-character `string`

默认值：

Use this character instead of a space to wrap long names.
设置盾标文本的折行字符。
* * *

shield-halo-fill `color`

默认值： #FFFFFF
(white)
白色

Specifies the color of the halo around the text.
设置盾标文本的光晕颜色。
* * *

shield-halo-radius `float`

默认值： 0
(no halo)
盾标文本无光晕效果

Specify the radius of the halo in pixels
设置盾标文本光晕的大小，单位为像素。
* * *

shield-character-spacing `unsigned`

默认值： 0

Horizontal spacing between characters (in pixels). Currently works for point placement only, not line placement.
设置盾标文字的字间距。该属性目前仅适用于点要素上的盾标。
* * *

shield-line-spacing `unsigned`

默认值： undefined

Vertical spacing between lines of multiline labels (in pixels)
设置盾标文本中的行距。
* * *

shield-text-dx `float`

默认值： 0

Displace text within shield by fixed amount, in pixels, +/- along the X axis. A positive value will shift the text right
设置盾标文本的水平偏移量，以像素为单位。正值表示向右偏移。
* * *

shield-text-dy `float`

默认值： 0

Displace text within shield by fixed amount, in pixels, +/- along the Y axis. A positive value will shift the text down
设置盾标文本的垂直偏移量，以像素为单位。正值表示向下偏移。
* * *

shield-dx `float`

默认值： 0

Displace shield by fixed amount, in pixels, +/- along the X axis. A positive value will shift the text right
设置盾标本身的水平偏移量，以像素为单位。正值表示向右偏移。
* * *

shield-dy `float`

默认值： 0

Displace shield by fixed amount, in pixels, +/- along the Y axis. A positive value will shift the text down
设置盾标本身的垂直偏移量，以像素为单位。正值表示向下偏移。
* * *

shield-opacity `float`

默认值： 1

The opacity of the image used for the shield
设置盾标背景图片的透明度。
* * *

shield-text-opacity `float`

默认值： 1

The opacity of the text placed on top of the shield
设置盾标文本的透明度。
* * *

shield-horizontal-alignment `keyword`

`left``middle``right``auto`

默认值： auto

The shield's horizontal alignment from its centerpoint
设置盾标相对于其中心点的水平对齐方式。
* * *

shield-vertical-alignment `keyword`

`top``middle``bottom``auto`

默认值： middle

The shield's vertical alignment from its centerpoint
设置盾标相对于其中心点的垂直对齐方式。
* * *

shield-placement-type `keyword`

`dummy``simple`

默认值： dummy

Re-position and/or re-size shield to avoid overlaps. "simple" for basic algorithm (using shield-placements string,) "dummy" to turn this feature off.
设置盾标之间相互避让的算法。simple表示使用由shield-placements属性指定的基本算法。而dummy则表示不使用该特性。
* * *

shield-placements `string`

默认值：

If "placement-type" is set to "simple", use this "POSITIONS,SIZES" string. An example is `shield-placements: &quot;E,NE,SE,W,NW,SW&quot;;`

如果shield-placement-type属性被设置为simple，那么就会依据该属性的值（即形如“POSITIONS, [SIZES]”的字符串）执行盾标相互避让算法。例如： `shield-placements: "E,NE,SE,W,NW,SW";`
* * *

shield-text-transform `keyword`

`none``uppercase``lowercase``capitalize`

默认值： none

Transform the case of the characters
设置是否对盾标字符进行大小写转换。
* * *

shield-justify-alignment `keyword`

`left``center``right``auto`

默认值： auto

Define how text in a shield's label is justified
设置盾标文本的对齐方式。
* * *

shield-transform `functions`

默认值：
(No transformation)

SVG transformation definition
设置SVG的变换函数。
* * *

shield-clip `boolean`

默认值： true
(geometry will be clipped to map bounds before rendering)

几何要素会根据地图的地理范围进行切割。

geometries are clipped to map bounds by default for best rendering performance. In some cases users may wish to disable this to avoid rendering artifacts.

为了提高绘制效率，可以先将矢量要素中所有超出地图边界的部分切掉，再进行绘制。但在某些情况下，为了防止出现绘制错误，也可以通过将该值设为false而不采用这个策略。
* * *

shield-comp-op `keyword`

`clear``src``dst``src-over``dst-over``src-in``dst-in``src-out``dst-out``src-atop``dst-atop``xor``plus``minus``multiply``screen``overlay``darken``lighten``color-dodge``color-burn``hard-light``soft-light``difference``exclusion``contrast``invert``invert-rgb``grain-merge``grain-extract``hue``saturation``color``value`

默认值： src-over
(add the current symbolizer on top of other symbolizer)

将当前符号置于其它符号的上一层

Composite operation. This defines how this symbolizer should behave relative to symbolizers atop or below it.

这也是一个合成操作。它定义了当前的符号应该如何与其相邻图层进行合成。

线模式（line-pattern）的属性

line-pattern-file `uri`

默认值： none

An image file to be repeated and warped along a line
设置沿线重复绘制的图像文件。
* * *

line-pattern-clip `boolean`

默认值： true
(geometry will be clipped to map bounds before rendering)

几何要素会根据地图的地理范围进行切割。

geometries are clipped to map bounds by default for best rendering performance. In some cases users may wish to disable this to avoid rendering artifacts.

为了提高绘制效率，可以先将矢量要素中所有超出地图边界的部分切掉，再进行绘制。但在某些情况下，为了防止出现绘制错误，也可以通过将该值设为false而不采用这个策略。
* * *

line-pattern-simplify `float`

默认值： 0
(geometry will not be simplified)
不对几何要素进行简化

geometries are simplified by the given tolerance
如果要对几何要素按照地图综合的方法进行简化，那么通过该属性来设定阈值。

* * *

line-pattern-simplify-algorithm `keyword`

`radial-distance``zhao-saalfeld``visvalingam-whyatt`

默认值： radial-distance
(geometry will not be simplified using the radial distance algorithm)

不使用radial-distance算法进行简化

geometries are simplified by the given algorithm

设置对线要素进行综合的简化算法。
* * *

line-pattern-smooth `float`

默认值： 0
(no smoothing)
不进行平滑处理

Range: 0-1
取值范围：0到1

Smooths out geometry angles. 0 is no smoothing, 1 is fully smoothed. Values greater than 1 will produce wild, looping geometries.
对线的拐点进行平滑处理。0表示不进行平滑，1表示完全平滑。如果取值大于1，会导致绘制的几何要素扭曲变形。

---

line-pattern-offset `float`

默认值： 0
(no offset)
无偏移

Offsets a line a number of pixels parallel to its actual path. Positive values move the line left, negative values move it right (relative to the directionality of the line).
将线要素相对于其原有位置向左（沿着线的走向）或向右偏移一定量的像素绘制。正值表示左偏，负值表示右偏。

---

line-pattern-geometry-transform `functions`

默认值： none
(geometry will not be transformed)
不对几何要素进行变换

Allows transformation functions to be applied to the geometry.
为几何要素定义变换函数。
* * *

line-pattern-comp-op `keyword`

`clear``src``dst``src-over``dst-over``src-in``dst-in``src-out``dst-out``src-atop``dst-atop``xor``plus``minus``multiply``screen``overlay``darken``lighten``color-dodge``color-burn``hard-light``soft-light``difference``exclusion``contrast``invert``invert-rgb``grain-merge``grain-extract``hue``saturation``color``value`

默认值： src-over
(add the current symbolizer on top of other symbolizer)
将当前符号置于其它符号的上一层

Composite operation. This defines how this symbolizer should behave relative to symbolizers atop or below it.
这也是一个合成操作。它定义了当前的符号应该如何与其相邻图层进行合成。
* * *

面模式（polygon-pattern）的属性

polygon-pattern-file `uri`

默认值： none

Image to use as a repeated pattern fill within a polygon
设置用于平铺填充面要素的图像文件。
* * *

polygon-pattern-alignment `keyword`

`local``global`

默认值： local

Specify whether to align pattern fills to the layer or to the map.
设置填充时的对齐方式，local指在当前图层中对齐，global指在整个地图中对齐。
* * *

polygon-pattern-gamma `float`

默认值： 1
(fully antialiased)
完全抗锯齿
Range: 0-1
Level of antialiasing of polygon pattern edges
设置面要素边缘的抗锯齿级别，影响绘制效果和速度。
* * *

polygon-pattern-opacity `float`

默认值： 1
(The image is rendered without modifications)
保持填充图片的透明度不变

Apply an opacity level to the image used for the pattern
设置填充图片的透明度。
* * *

polygon-pattern-clip `boolean`

默认值： true
(geometry will be clipped to map bounds before rendering)
几何要素会根据地图的地理范围进行切割

geometries are clipped to map bounds by default for best rendering performance. In some cases users may wish to disable this to avoid rendering artifacts.
为了提高绘制效率，可以先将矢量要素中所有超出地图边界的部分切掉，再进行绘制。但在某些情况下，为了防止出现绘制错误，也可以通过将该值设为false而不采用这个策略。

---

polygon-pattern-simplify `float`

默认值： 0
(geometry will not be simplified)
不对面要素的边线进行简化

geometries are simplified by the given tolerance
如果要对面要素的边线按照地图综合的方法进行简化，那么通过该属性来设定阈值。
* * *

polygon-pattern-simplify-algorithm `keyword`

`radial-distance``zhao-saalfeld``visvalingam-whyatt`

默认值： radial-distance
(geometry will not be simplified using the radial distance algorithm)
不使用radial-distance算法进行简化（？）

geometries are simplified by the given algorithm
设置对面要素的边线进行综合的简化算法。
* * *

polygon-pattern-smooth `float`

默认值： 0
(no smoothing)
不对拐点进行平滑

Range: 0-1
取值范围：0到1

Smooths out geometry angles. 0 is no smoothing, 1 is fully smoothed. Values greater than 1 will produce wild, looping geometries.
对线的拐点进行平滑处理。0表示不进行平滑，1表示完全平滑。如果取值大于1，会导致绘制的几何要素扭曲变形。
* * *

polygon-pattern-geometry-transform `functions`

默认值： none
(geometry will not be transformed)
不对几何要素进行变换。

Allows transformation functions to be applied to the geometry.
为几何要素定义变换函数。
* * *

polygon-pattern-comp-op `keyword`

`clear``src``dst``src-over``dst-over``src-in``dst-in``src-out``dst-out``src-atop``dst-atop``xor``plus``minus``multiply``screen``overlay``darken``lighten``color-dodge``color-burn``hard-light``soft-light``difference``exclusion``contrast``invert``invert-rgb``grain-merge``grain-extract``hue``saturation``color``value`

默认值： src-over
(add the current symbolizer on top of other symbolizer)
将当前符号置于其它符号的上一层

Composite operation. This defines how this symbolizer should behave relative to symbolizers atop or below it.
这也是一个合成操作。它定义了当前的符号应该如何与其相邻图层进行合成。

* * *

栅格符号（raster）的属性

raster-opacity `float`

默认值： 1
(opaque)
不透明

The opacity of the raster symbolizer on top of other symbolizers.
设置栅格符号的透明度。
* * *

raster-filter-factor `float`

默认值： -1
(Allow the datasource to choose appropriate downscaling.)
允许数据源自行选用缩小图像尺寸（重采样？）的方法

This is used by the Raster or Gdal datasources to pre-downscale images using overviews. Higher numbers can sometimes cause much better scaled image output, at the cost of speed.
用于栅格或GDAL数据源（译注：这个是mapnik概念），对图像尺寸进行预先缩小。将该值调高可以得到更好（译注：什么叫“好”？）的缩略图效果，但相应的处理时间也会变长。
* * *

raster-scaling `keyword`

`near``fast``bilinear``bilinear8``bicubic``spline16``spline36``hanning``hamming``hermite``kaiser``quadric``catrom``gaussian``bessel``mitchell``sinc``lanczos``blackman`

默认值： near

The scaling algorithm used to making different resolution versions of this raster layer. Bilinear is a good compromise between speed and accuracy, while lanczos gives the highest quality.
设置对栅格数据进行重采样的算法。Bilinear可以在速度和质量方面得到不错的平衡，而lanczos则能够得到最高的绘制质量。
* * *

raster-mesh-size `unsigned`

默认值： 16
(Reprojection mesh will be 1/16 of the resolution of the source image)
取原始图像分辨率的1/16作为栅格的重投影网格大小（译注：对于256*256的瓦片，网格的大小就是256/16=16，也就是16*16这么大。不知道这么理解对不对，不过至少从之前绘制过的栅格，特别是出现那些奇怪小网格的栅格数据情况来看，应该是这样的。）

A reduced resolution mesh is used for raster reprojection, and the total image size is divided by the mesh-size to determine the quality of that mesh. Values for mesh-size larger than the default will result in faster reprojection but might lead to distortion.
在对原始图像进行重投影时，是先将图像切分成若干网格，对网格中的小图像片分别重投影。如果设定该值使得网格的尺寸变大（译注：即把该属性的值调小），那么重投影的速度会加快，但可能会导致图像变形。
* * *

raster-comp-op `keyword`

`clear``src``dst``src-over``dst-over``src-in``dst-in``src-out``dst-out``src-atop``dst-atop``xor``plus``minus``multiply``screen``overlay``darken``lighten``color-dodge``color-burn``hard-light``soft-light``difference``exclusion``contrast``invert``invert-rgb``grain-merge``grain-extract``hue``saturation``color``value`

默认值： src-over
(add the current symbolizer on top of other symbolizer)
将当前符号置于其它符号的上一层

Composite operation. This defines how this symbolizer should behave relative to symbolizers atop or below it.
这也是一个合成操作。它定义了当前的符号应该如何与其相邻图层进行合成。
* * *

raster-colorizer-default-mode `keyword`

`discrete``linear``exact`

默认值： undefined

TODO
* * *

raster-colorizer-default-color `color`

默认值： undefined

TODO
* * *

raster-colorizer-epsilon `float`

默认值： undefined

TODO
* * *

raster-colorizer-stops `tags`

默认值： undefined

TODO
* * *

注记符号（markers）的属性

marker-file `uri`

默认值:
(An ellipse or circle, if width equals height)

一个椭圆或正圆形符号

An SVG file that this marker shows at each placement. If no file is given, the marker will show an ellipse.

设置绘制注记符号所使用的SVG文件。如果没有指定具体的文件，则默认使用一个椭圆形符号对每个位置的注记进行渲染。

---

marker-opacity `float`

默认值： 1
(The stroke-opacity and fill-opacity will be used)

边缘和填充均不透明

The overall opacity of the marker, if set, overrides both the opacity of both the fill and stroke

设置注记符号整体的透明度。如果设置了该属性，则会覆盖在marker-fill-opacity和marker-line-opacity属性中设置的透明度。

---

marker-fill-opacity `float`

默认值： 1
(opaque)

注记符号填充部分为不透明

The fill opacity of the marker

设置注记符号填充部分的透明度。

---

marker-line-color `color`

默认值： black

The color of the stroke around a marker shape.

设置注记符号边线的颜色。

---

marker-line-width `float`

默认值： undefined

The width of the stroke around a marker shape, in pixels. This is positioned on the boundary, so high values can cover the area itself.

设置注记符号边线的宽度，以像素为单位。但如果该值设置过大会导致注记本身被过粗的边线覆盖。

---

marker-line-opacity `float`

默认值： 1
(opaque)

注记符号边线完全不透明

The opacity of a line

设置注记符号边线的透明度。
* * *

marker-placement `keyword`

`point``line``interior`

默认值： point
(Place markers at the center point (centroid) of the geometry)

注记符号被置于几何要素的重心（形心）位置

Attempt to place markers on a point, in the center of a polygon, or if markers-placement:line, then multiple times along a line. 'interior' placement can be used to ensure that points placed on polygons are forced to be inside the polygon interior

设置注记在几何要素上的放置方式，可以位于点要素上，或者在面要素的中心位置，还可以沿着线要素反复出现（通过设置markers-placement:line实现）。如果取值interior，那么可以确保注记符号被绘制在多边形的内部。

---

marker-multi-policy `keyword`

`each``whole``largest`

默认值： each
(If a feature contains multiple geometries and the placement type is either point or interior then a marker will be rendered for each)

如果一个要素包含了多个几何形状，而且放置方式是point或interior，那么注记符号就会在每个几何形状处都会被绘制一次

A special setting to allow the user to control rendering behavior for 'multi-geometries' (when a feature contains multiple geometries). This setting does not apply to markers placed along lines. The 'each' policy is default and means all geometries will get a marker. The 'whole' policy means that the aggregate centroid between all geometries will be used. The 'largest' policy means that only the largest (by bounding box areas) feature will get a rendered marker (this is how text labeling behaves by default).

该属性是为包含多个几何形状的地理（multi-geometries）要素准备的，对沿线放置的注记符号不起作用。其默认值为each，也就是每个几何形状上都会被绘制一个注记；whole表示注记将被绘制在所有几何形状组合后的重心位置；而largest表示注记将被绘制在最大（依据最小包围框的面积）的那个几何形状上（这也同样是文本标注在多几何要素上绘制的默认方法）。

---

marker-type `keyword`

`arrow``ellipse`

默认值： ellipse

The default marker-type. If a SVG file is not given as the marker-file parameter, the renderer provides either an arrow or an ellipse (a circle if height is equal to width)

设置默认的注记符号类型。如果没有指定用于渲染注记的SVG文件，那么内置的渲染引擎可以提供两种选择：箭头或椭圆。

---

marker-width `expression`

默认值： 10

The width of the marker, if using one of the default types.

设置注记符号的宽度。这个属性只适用于两种内置的默认注记样式。
* * *

marker-height `expression`

默认值： 10

The height of the marker, if using one of the default types.

设置注记符号的高度。这个属性只适用于两种内置的默认注记样式。

---

marker-fill `color`

默认值： blue

The color of the area of the marker.

设置注记的填充色。

---

marker-allow-overlap `boolean`

默认值： false
(Do not allow makers to overlap with each other - overlapping markers will not be shown.)

不允许注记符号相互压盖（被压盖的注记将不被显式）

Control whether overlapping markers are shown or hidden.

设置被压盖的注记符号是否被显式在地图上。

---

marker-ignore-placement `boolean`

默认值： false
(do not store the bbox of this geometry in the collision detector cache)

不在冲突检测器的缓存中存储几何形状的外包框

value to control whether the placement of the feature will prevent the placement of other features

设置是否允许在与当前要素重叠的位置放置其它要素。（？）

---

marker-spacing `float`

默认值： 100

Space between repeated markers in pixels. If the spacing is less than the marker size or larger than the line segment length then no marker will be placed

设置重复绘制的注记之间的间距，单位为像素。如果设定的间距小于注记符号本身的尺寸，或者大于线要素的长度，那么注记就绘制不出来。

---

marker-max-error `float`

默认值： 0.2

The maximum difference between actual marker placement and the marker-spacing parameter. Setting a high value can allow the renderer to try to resolve placement conflicts with other symbolizers.

设置实际的注记位置与marker-spacing属性值之间的最大误差。如果将该属性值调高，那么渲染引擎就会尝试处理与其它注记符号之间的位置冲突。

---

marker-transform `functions`

默认值:
(No transformation)

无变换

SVG transformation definition

设置SVG图形的变换方法
* * *

marker-clip `boolean`

默认值： true
(geometry will be clipped to map bounds before rendering)

几何要素会根据地图的地理范围进行切割。

geometries are clipped to map bounds by default for best rendering performance. In some cases users may wish to disable this to avoid rendering artifacts.

为了提高绘制效率，可以先将矢量要素中所有超出地图边界的部分切掉，再进行绘制。但在某些情况下，为了防止出现绘制错误，也可以通过将该值设为false而不采用这个策略。
* * *

marker-smooth `float`

默认值： 0
(no smoothing)
不对拐点进行平滑

Range: 0-1
取值范围：0到1

Smooths out geometry angles. 0 is no smoothing, 1 is fully smoothed. Values greater than 1 will produce wild, looping geometries.
对线的拐点进行平滑处理。0表示不进行平滑，1表示完全平滑。如果取值大于1，会导致绘制的几何要素扭曲变形。
* * *

marker-geometry-transform `functions`

默认值： none
(geometry will not be transformed)
不对几何要素进行变换

Allows transformation functions to be applied to the geometry.
为几何要素定义变换函数。
* * *

marker-comp-op `keyword`

`clear``src``dst``src-over``dst-over``src-in``dst-in``src-out``dst-out``src-atop``dst-atop``xor``plus``minus``multiply``screen``overlay``darken``lighten``color-dodge``color-burn``hard-light``soft-light``difference``exclusion``contrast``invert``invert-rgb``grain-merge``grain-extract``hue``saturation``color``value`

默认值： src-over
(add the current symbolizer on top of other symbolizer)
将当前符号置于其它符号的上一层

Composite operation. This defines how this symbolizer should behave relative to symbolizers atop or below it.
这也是一个合成操作。它定义了当前的符号应该如何与其相邻图层进行合成。
* * *

建筑物符号（building）的属性

building-fill `color`

默认值： #FFFFFF
(白色)

The color of the buildings walls.
设置建筑物的外墙填充色

* * *

building-fill-opacity `float`

默认值： 1

The opacity of the building as a whole, including all walls.
设置建筑物整体的透明度，包括建筑物所有的面。
* * *

building-height `expression`

默认值： 0

The height of the building in pixels.
设置建筑物的高度，以像素为单位。
* * *

调试模式下的属性

debug-mode `string`

默认值： collision

The mode for debug rendering
设置调试模式渲染（？）
* * *

关于取值类型的说明

Below is a list of values and an explanation of any expression that can be applied to properties in CartCSS.

这里列出了CartoCSS中所有属性的取值类型及其说明。

颜色型（Color）

CartoCSS accepts a variety of syntaxes for colors - HTML-style hex values, rgb, rgba, hsl, and hsla. It also supports the predefined HTML colors names, like `yellow` and `blue`.

CartoCSS可以使用一系列不同的方法表示颜色：HTML风格的16进制值，RGB值，RGBA值，HSL值或HSLA值都可以，还可以使用HTML预定义颜色名，像`yellow`、`blue`等。

```
#line {
 line-color: #ff0;
 line-color: #ffff00;
 line-color: rgb(255, 255, 0);
 line-color: rgba(255, 255, 0, 1);
 line-color: hsl(100, 50%, 50%);
 line-color: hsla(100, 50%, 50%, 1);
 line-color: yellow;
}
```

Especially of note is the support for hsl, which can be easier to reason about than rgb()(http://mothereffinghsl.com/). Carto also includes several color functions borrowed from less(http://lesscss.org/#-color-functions):

这里特别需要强调的是对HSL值的支持，这其实是比RGB值更易用的颜色表达方式（参见这里）。CartoCSS中还支持几种颜色函数，这是从LESS中借用的概念（参见这里），例子如下：

```
// 把颜色调亮或调暗
lighten(#ace, 10%);
darken(#ace, 10%);

// 饱和度调高或调低
saturate(#550000, 10%);
desaturate(#00ff00, 10%);

// 提高或降低颜色的透明度
fadein(#fafafa, 10%);
fadeout(#fefefe, 14%);

// 按照一定角度旋转色盘
spin(#ff00ff, 10);

// 将两种颜色混合
mix(#fff, #000, 50%);
```

These functions all take arguments which can be color variables, literal colors, or the results of other functions operating on colors.

以上这些函数的参数可以是颜色值，也可以是颜色名，还可以是其它颜色函数。

浮点型（Float）

Float is a fancy way of saying 'number'. In CartoCSS, you specify *just a number* - unlike CSS, there are no units, but everything is specified in pixels.

浮点数是数值类型的时髦说法。在CartoCSS中，这指的就是一个数值，没有单位，但其实所有的单位都是像素。

```
#line {
 line-width: 2;
```

```
}
```

It's also possible to do simple math with number values:
还可以对数值类型做简单运算：

```
#line {
 line-width: 4 / 2; // 除
 line-width: 4 + 2; // 加
 line-width: 4 - 2; // 减
 line-width: 4 * 2; // 乘
 line-width: 4 % 2; // 取余
}
```

统一资源描述符型（URI）

URI is a fancy way of saying URL. When an argument is a URI, you use the same kind of `url('place.png')` notation that you would with HTML. Quotes around the URL aren't required, but are highly recommended. URIs can be paths to places on your computer, or on the internet.

URI是URL的一种时髦说法（译注：这实在不敢苟同，在http协议和REST架构中，URI和URL都有明确的定义，它们是**不同的**）。当一个属性的值类型是URI时，用户可以像在HTML中使用`url('place.png')`一样的表示方法。URL地址上的引号不是必需的，但最好加上。URI可以指向本地文件系统，也可以是互联网上资源的链接地址。

```
#markers {
 marker-file: url('marker.png');
}
```

字符串型（String）

A string is basically just text. In the case of CartoCSS, you're going to put it in quotes. Strings can be anything, though pay attention to the cases of `text-name` and `shield-name` - they actually will refer to features, which you refer to by putting them in brackets, as seen in the example below.

字符串也就是文本类型。在CartoCSS中，字符串应该有引号包围。字符串可以是任意文本，但在`text-name`和`shield-name`属性中，可以使用中括号包围的数据字段名来表示。例如：

```
#labels {
 text-name: "[MY_FIELD]";
}
```

布尔型（Boolean）

Boolean means yes or no, so it accepts the values `true` or `false`.
布尔类型即是或否，取值为`true`或`false`。

```
#markers {
 marker-allow-overlap:true;
}
```

表达式型（Expressions）

Expressions are statements that can include fields, numbers, and other types in a really flexible way. You have run into expressions before, in the realm of 'fields', where you'd specify `"[FIELD]"`, but expressions allow you to drop the quotes and also do quick addition, division, multiplication, and concatenation from within Carto syntax.

表达式是一种语句，它可以将数据字段、数值以及其它类型灵活的组合起来。前面提到的`"[FIELD]"`形式包含了表达式。实际的表达式可以不用加引号就执行加、减、乘、除、连接等CartoCSS语法支持的操作。

```
#buildings {
 building-height: [HEIGHT_FIELD] * 10;
}
```

数列型（Numbers）

Numbers are comma-separated lists of one or more number in a specific order. They're used in line dash arrays, in which the numbers specify intervals of line, break, and line again.
数列型是逗号分隔的一组有序数值。数列类型在用于配置虚线样式时，其中的数字交替表示的是实线段长度、间隔长度和实线段长度。

```
#disputedboundary {
 line-dasharray: 1, 4, 2;
}
```

百分数型（Percentages）

In Carto, the percentage symbol, `%` universally means `value/100`. It's meant to be used with ratio-related properties, like opacity rules.
在CartoCSS中，百分号`%`表示`值/100`。它可以用于表示比例的属性，例如透明度。

You should not use percentages as widths, heights, or other properties - unlike CSS, percentages are not relative to cascaded classes or page size, they're, as stated, simply the

value divided by one hundred.

注意，百分数不能用于定义宽度、高度等属性。这一点与CSS不同，因为在CartoCSS中没有CSS中层次化的页面要素和页宽。它们在这里只是除以100以后的值。

```
#world {
 // 这种表达方式与...
 polygon-opacity: 50%;

 // ...这种方式效果一样
 polygon-opacity: 0.5;
}
```

函数型（Functions）

Functions are comma-separated lists of one or more functions. For instance, transforms use the `functions` type to allow for transforms within Carto, which are optionally chainable.
这种类型可以包含一组逗号分隔的函数。例如，各种变换都是用 `functions` 作为值类型，而且这些函数还可以串接起来（？）。

```
#point {
 point-transform: scale(2, 2);
}
```