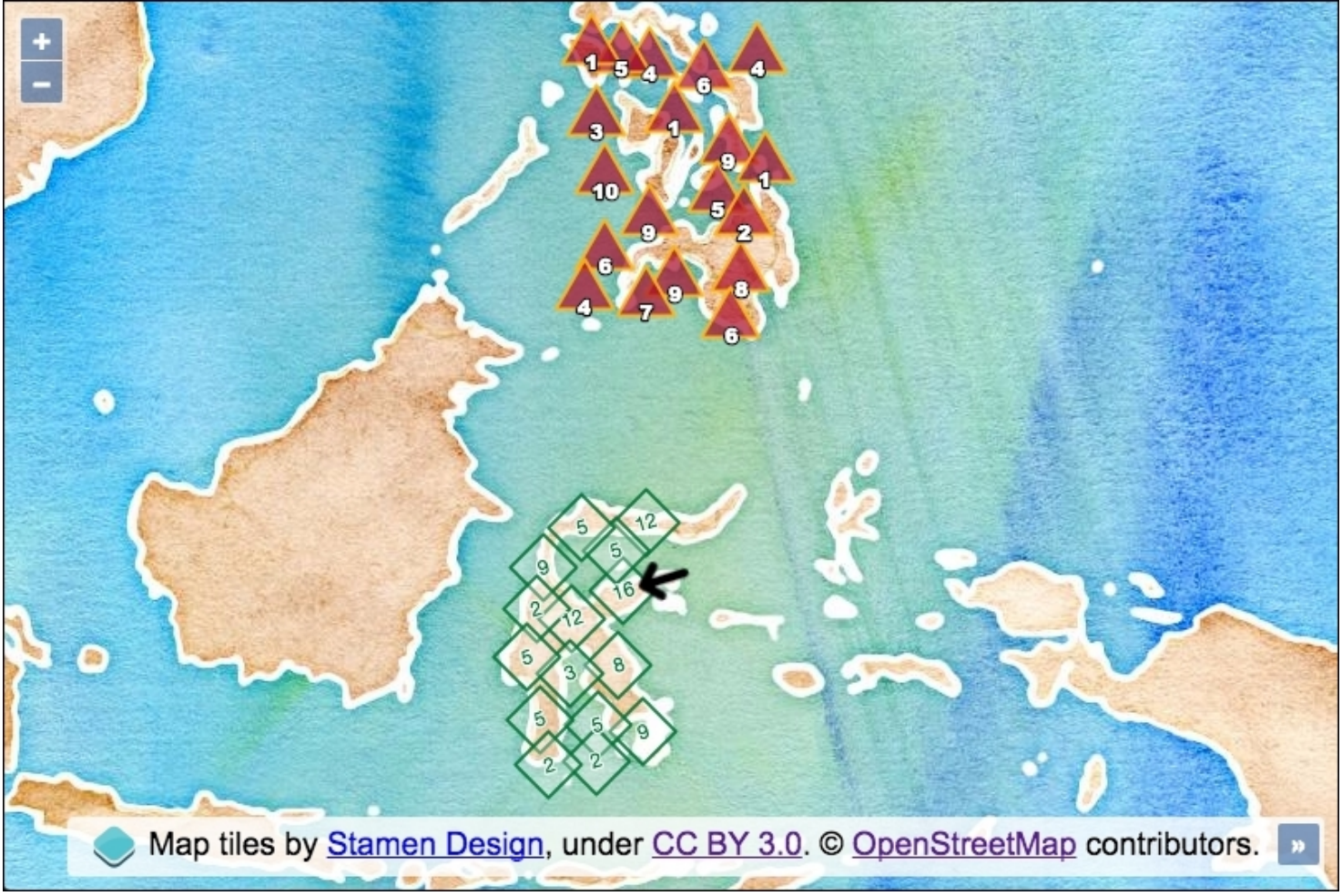
样式化群集特征

什么时候在处理许多特征点时，通常使用聚类策略来避免点重叠并提高渲染性能。我们之前已经详细介绍集群战略在**使用集群战略**在配方第3章 (/book/web_development/9781785287756/3)，**工作与矢量图层**，虽然我们在配方引入了一些自定义的样式，我们将看到更多样式选项，以增强外观此食谱中的群集。

我们将加载两个单独的GeoJSON文件，这些文件包含大约100个点的几何图形，每个文件都位于一个集中的区域，以显示聚类策略为此类数据提供的好处。我们将在不同的图层上渲染这两套要素，每层都有独特的样式。

另外，对于其中一个矢量层，我们将包括一个箭头图标，该箭头指向在其下方具有大量点的聚类。通过查看以下屏幕截图，您可以了解我们的意思。

来源可以在中找到代码 `ch06/ch06-styling-clustered-features` ，如下所示（如您所见，集群可以完全转换为最适合您的应用程序的样式）：



怎么做...

为了创建可以补充您自己的特定地图应用程序的独特样式的群集，请执行以下步骤：

1 创建一个具有OpenLayers依赖项的HTML文件以及一个 `div` 用于保存地图的元素。

2 在自定义JavaScript文件中，实例化的实例 `ol.Map` ：

复制

```
var map = new ol.Map({
  view: new ol.View({
    zoom: 5, center: [13565432, -577252]
  }),
  target: 'js-map',
  layers: [
    new ol.layer.Tile({
      source: new ol.source.Stamen({layer: 'watercolor'})
    })
  ]
});
```

3 创建 第一个矢量层样式函数，如下所示：

复制

```
var style1 = function(feature) {
  var length = feature.get('features').length;
  var styles = [
    new ol.style.Style({
      image: new ol.style.RegularShape({
        radius: 20, points: 4,
        stroke: new ol.style.Stroke({
          color: [1, 115, 54, 0.9], width: 2
        }),
        fill: new ol.style.Fill({
          color: [255, 255, 255, 0.3]
        })
      }),
      text: new ol.style.Text({
        text: length.toString(), rotation: -0.3,
        font: '12px "Helvetica Neue", Arial',
        fill: new ol.style.Fill({
          color: [1, 115, 54, 1]
        }),
```

4 创建 第二个矢量层样式函数：

复制

```
var style2 = function(feature) {
  var length = feature.get('features').length;
  return [
    new ol.style.Style({
      image: new ol.style.RegularShape({
        radius: 20, points: 3,
        stroke: new ol.style.Stroke({
          color: [255, 188, 17, 0.9], width: 2
        }),
        fill: new ol.style.Fill({
          color: [173, 10, 43, 0.7]
        })
      }),
      text: new ol.style.Text({
        text: length.toString(), offsetY: 9,
        font: '12px "Arial Black"',
        fill: new ol.style.Fill({
          color: 'white'
        }),
```

5 设置可以实例化新的群集矢量层的方法：

复制

```
var createClusteredLayer = function(url, style) {
  return new ol.layer.Vector({
    source: new ol.source.Cluster({
      distance: 25, projection: 'EPSG:3857',
      source: new ol.source.Vector({
        url: url,
        format: new ol.format.GeoJSON({
          defaultDataProjection: 'EPSG:3857'
        })
      })
    }),
    style: style
  })
};
```

6 最后，添加 具有不同源URL和样式功能的两个群集矢量层：

复制

```
map.addLayer(createClusteredLayer('points1.geojson', style1));
map.addLayer(createClusteredLayer('points2.geojson', style2));
```

怎么运行的...

此处的许多代码对于本章前面的食谱应该看起来很熟悉。让我们仔细看一下本食谱中实现的一些新使用的属性和自定义JavaScript。

我们将首先分解一下 `style1` 函数的类型 `ol.style.StyleFunction` （我们在前面的**基于要素属性**的“**样式”中**看到的一种**样式**方法）：

复制

```
var style1 = function(feature) {
  var length = feature.get('features').length;
  var styles = [
    new ol.style.Style({
      image: new ol.style.RegularShape({
        radius: 20, points: 4,
        stroke: new ol.style.Stroke({
          color: [1, 115, 54, 0.9], width: 2
        }),
```

我们将特征长度（簇数）的参考存储在一个变量中，即 `length` 。这就是用于文本标签的内容。

我们创建的新实例， `ol.style.Style` 并为该属性分配一个自定义 `ol.style.RegularShape` 实例 `image` 。我们使用这种类型的样式，以便**根据几何类型**配方为**样式特征**创建星形几何。但是，这一次我们使用它来创建正方形几何体（ `points` 分配为 `4` “多边形边数”）。

复制

```
text: new ol.style.Text({
  text: length.toString(), rotation: -0.3,
```

该片段也从 `style1` 功能中删除。我们可以看到，当实例化一个 `ol.style.Text` （针对集群标签的）新实例时，我们使用了 `length` 变量的计数，请确保将其转换为字符串类型，并通过该 `rotation` 属性将文本旋转设置为非默认值。该值以弧度为单位，并产生倾斜效果。

复制

```
if (length > 15) {
    styles.push(new ol.style.Style({
      image: new ol.style.Icon({
        src: '../../assets/images/arrow-left.png',
        rotation: -0.3, scale: 1.2,
        anchor: [-0.2, 0.5]
      })
    }))
  }
```

最后一个专业该 `style1` 功能的一部分是先前对集群的功能计数进行条件检查。

如果群集中的要素数量大于15，则 `ol.style.Style` 在返回完成的styles数组之前，将数组的另一个实例推送到数组中。我们实例化了一个实例， `ol.style.Icon` 以向该群集功能提供一个箭头图标。就像文本标签一样，我们将图标以 `0.3` 弧度旋转，并且还将图标按其原始大小（ `1.2` ）放大20%。

该 `anchor` 属性包含使用小数单位的图标的**x，y**锚点（您可以通过 `anchorXUnits` 或 `anchorYUnits` 属性将其调整为使用像素单位）。默认值使用[0.5，0.5]的分数单位。我们对此进行了修改，以使箭头偏移到群集中心的右侧， `-0.2` 而不是 `0.5` 。

让我们继续研究一下该 `style2` 函数，方法是只抽取一部分方法进行讨论：

复制

```
image: new ol.style.RegularShape({
  radius: 20, points: 3,
```

同样，我们使用的实例来自 `ol.style.RegularShape` 定义用于集群功能的几何。为 `points` 属性分配边的值后 `3` ，它将生成一个三角形。

复制

```
text: new ol.style.Text({
  text: length.toString(), offsetY: 9,
  font: '12px "Arial Black"',
```

此代码段是 `style2` 我们要查看的功能的最后一个代码段。它创建文本样式构造函数的实例。新引入的属性是 `offsetY` ，用于将文本标签（群集特征数）垂直对齐到三角形的底部。所使用的排版是较粗的字体（ `Arial Black` ）。

复制

```
var createClusteredLayer = function(url, style) {
  return new ol.layer.Vector({
    source: new ol.source.Cluster({
      distance: 25, projection: 'EPSG:3857',
      source: new ol.source.Vector({
        url: url
        format: new ol.format.GeoJSON({
          defaultDataProjection: 'EPSG:3857'
        })
      })
    }),
    style: style
  })
};
```

我们创建了一个函数返回矢量层的自定义实例。该函数接受 `url` 源URL 的参数以及 `style` 该图层要使用的样式函数的参数。

我们通过属性 `EPSG:3857` 在群集实例（ `ol.source.Cluster` ）上指定了投影 `projection` ，并且在通过 `defaultDataProjection` 属性格式化GeoJSON时也明确了该投影。

样式功能应用于矢量图层 `style` 属性，这意味着图层样式规则将应用于图层中的每个要素。

也可以看看

- 该**造型交互渲染意图**食谱
- **基于几何类型**配方的**样式功能**
- 第3章 (/book/web_development/9781785287756/3)，**使用向量层中**的**使用群集策略**食谱 (/book/web_development/9781785287756/3)

上一节 (/book/web_development/9781785287756/6/ch06lvl1sec58/styling-interaction-render-i

下一节 (/book/web_development/9781785287756/7)