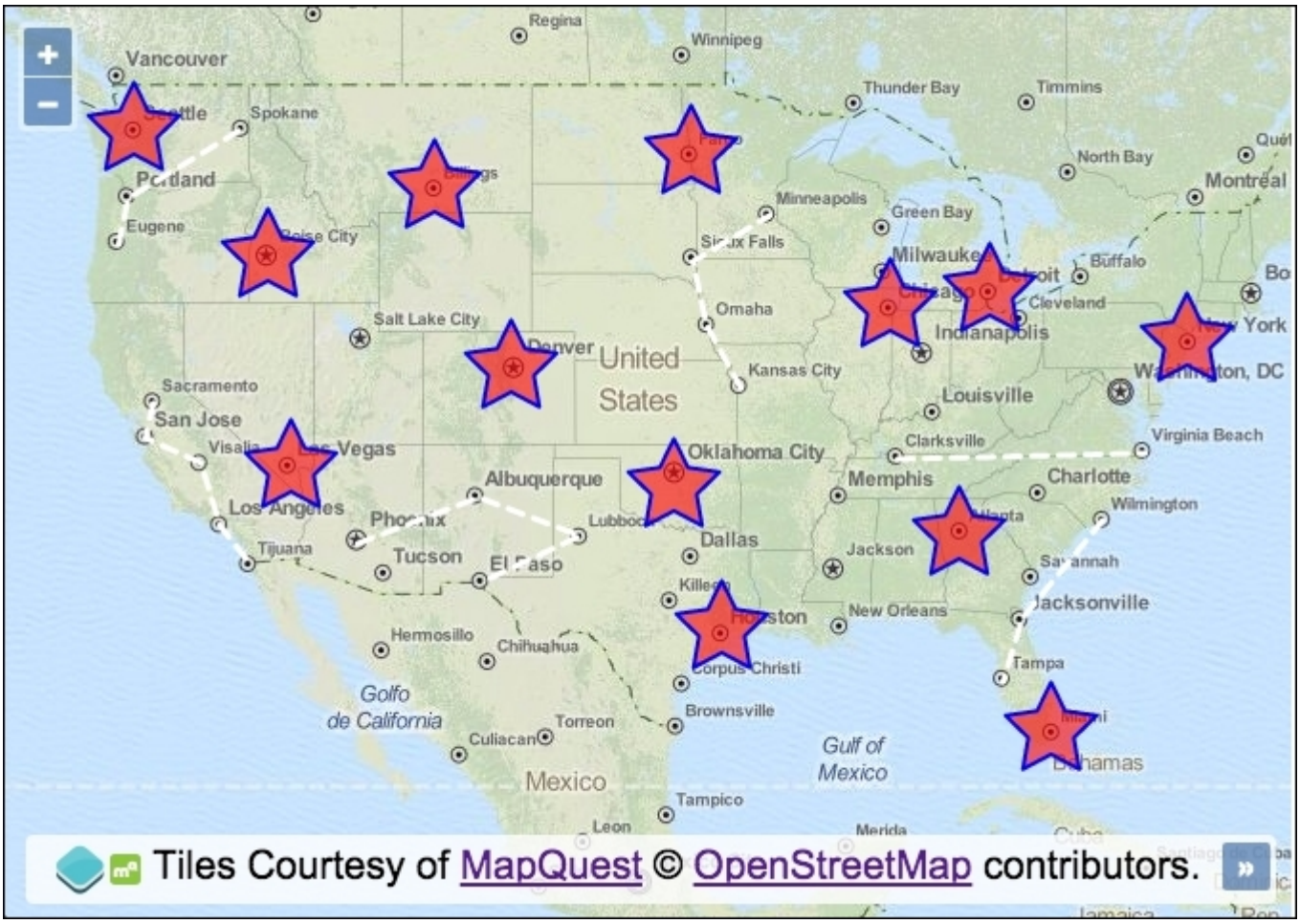
基于几何类型的样式特征

我们可以总结出样式化样式的方法有两种。首先是通过将样式应用于图层，以便每个要素都继承此样式，如“**样式图层**”配方中所示。第二种是将样式选项直接应用到功能，我们将在本食谱中看到。

此配方向我们展示了如何根据几何类型选择要应用于特征的样式风格。我们将使用 `ol.Feature` 方法将样式直接应用于要素 `setStyle` 。

当 `point` 检测到几何类型时，我们实际上将代表几何的样式设置为星形，而不是默认的圆形。当检测到线串的几何类型时，将应用其他样式。

可以在中找到源代码 `ch06/ch06-styling-features-by-geometry-type` ，配方的输出将类似于以下屏幕截图：



怎么做...

至 根据几何类型自定义要素样式，请使用以下步骤：

1 创建具有OpenLayers依赖项的HTML文件，jQuery库以及 `div` 将保存 `map` 实例的元素：

2 创建一个自定义JavaScript文件并初始化一个新 `map` 实例，如下所示：

复制

```
var map = new ol.Map({
  view: new ol.View({
    zoom: 4, center: [-10732981, 4676723]
  }),
  target: 'js-map',
  layers: [
    new ol.layer.Tile({
      source: new ol.source.MapQuest({layer: 'osm'})
    })
  ]
```

3 创建一个新的矢量层并将其添加到中 `map` 。让源加载器函数检索GeoJSON文件，格式化响应，然后将其传递到我们的自定义 `modifyFeatures` 方法（我们将在下一个实现），然后再将功能部件添加到矢量源：

复制

```
var vectorLayer = new ol.layer.Vector({
  source: new ol.source.Vector({
    loader: function() {
      $.ajax({
        type: 'GET',
        url: 'features.geojson',
        context: this
      }).done(function(data) {
        var format = new ol.format.GeoJSON();
        var features = format.readFeatures(data);
        this.addFeatures(modifyFeatures(features));
      });
    }
  })
});
map.addLayer(vectorLayer);
```

4 完 通过实现该 `modifyFeatures` 功能来关闭，从而 `geometry` 根据几何类型转换要素的投影并为其设置样式：

复制

```
function modifyFeatures(features) {
  features.forEach(function(feature) {
    var geometry = feature.getGeometry();
    geometry.transform('EPSG:4326', 'EPSG:3857');
    if (geometry.getType() === 'Point') {
      feature.setStyle(
        new ol.style.Style({
          image: new ol.style.RegularShape({
            fill: new ol.style.Fill({
              color: [255, 0, 0, 0.6]
            }),
            stroke: new ol.style.Stroke({
              width: 2, color: 'blue'
            }),
            points: 5, radius1: 25, radius2: 12.5
          })
        })
      );
    }
```

怎么运行的...

让我们来在 `loader` 仔细研究样式背后的逻辑之前，请简要了解向量源的功能：

复制

```
loader: function() {
  $.ajax({
    type: 'GET',
    url: 'features.geojson',
    context: this
  }).done(function(data) {
    var format = new ol.format.GeoJSON();
    var features = format.readFeatures(data);
    this.addFeatures(modifyFeatures(features));
  });
}
```

我们看到这种类型的源加载如何工作第3章 (/book/web_development/9781785287756/3)，**工作与矢量图层**的配方，**阅读功能直接使用AJAX**，但由于一些相对于这个配方方面在此提醒，我们将讨论的 `done` 承诺内容一旦收到AJAX响应。

我们的外部资源包含GeoJSON格式的点和线字符串，因此我们必须创建一个新实例，`ol.format.GeoJSON` 以便我们可以读取 `format.readFeatures(data)` AJAX响应的数据（）以构建OpenLayers功能的集合。

在将特征组直接添加到向量源（ `this` 此处指向量源）之前，我们通过我们的 `modifyFeatures` 方法传递特征数组。此方法将对所有要素应用所有必要的样式，然后将修改后的要素返回原位，并将结果输入该 `addFeatures` 方法。让我们分解一下 `modifyFeatures` 方法的内容：

复制

```
function modifyFeatures(features) {
  features.forEach(function(feature) {
    var geometry = feature.getGeometry();
    geometry.transform('EPSG:4326', 'EPSG:3857');
```

的逻辑首先使用JavaScript数组方法遍历数组中的每个功能 `forEach` 。传递给匿名迭代器函数的第一个参数是功能（ `feature` ）。

在循环迭代中，我们将此特征的几何存储在一个变量中，即 `geometry` ，因为在循环迭代期间多次访问了此要素。

您不知道，GeoJSON文件中的坐标投影是以经度/纬度，投影代码表示的 `EPSG:4326` 。但是，地图的视图处于投影状态 `EPSG:3857` 。为了确保它们出现在地图上预期的位置，我们使用了 `transform` 几何方法，该方法将源投影和目标投影作为参数，并在适当的位置转换几何的坐标。

复制

```
if (geometry.getType() === 'Point') {
  feature.setStyle(
    new ol.style.Style({
      image: new ol.style.RegularShape({
        fill: new ol.style.Fill({
          color: [255, 0, 0, 0.6]
        }),
        stroke: new ol.style.Stroke({
          width: 2, color: 'blue'
        }),
        points: 5, radius1: 25, radius2: 12.5
      })
    })
  );
}
```

接下来是有条件检查几何是否为的类型 `Point` 。该 `geometry` 实例有方法 `getType` 为这种目的。

内联实例 `setStyle` 方法 `feature` ，我们 `style` 从 `ol.style.Style` 构造函数创建一个新对象。我们感兴趣的唯一直接属性是该 `image` 属性。

默认情况下，点几何形状设置为圆形。相反，我们要将点设置为星形。我们可以通过使用 `ol.style.RegularShape` 构造函数来实现。我们 `fill` 使用 `color` 属性设置样式，并 `stroke` 使用 `width` 和 `color` 属性设置样式。

该 `points` 物业指定星星的点数。在多边形的情况下，它表示边数。

的 `radius1` 和 `radius2` 特性具体地，设计星形状为内和外半径的配置分别。

复制

```
if (geometry.getType() === 'LineString') {
  feature.setStyle(
    new ol.style.Style({
      stroke: new ol.style.Stroke({
        color:  [255, 255, 255, 1],
        width: 3, lineDash: [8, 6]
      })
    })
  );
}
```

该方法的最后一部分对的几何类型进行条件检查 `LineString` 。在这种情况下，我们将此几何类型设置为与点几何类型不同的样式。我们提供 `stroke` 带有 `color` , `width` 和自定义破折号的样式。该 `lineDash` 数组声明的行长为 `8` ，其后跟一个间隙长度为 `6` 。

也可以看看

- 该**造型交互渲染意图**食谱
- 该**造型层**配方
- 在**创建功能编程**配方在第3章 (/book/web_development/9781785287756/3)，**工作与矢量图层**

上一节 (/book/web_development/9781785287756/6/ch06lvl1sec55/styling-layers)

下一节 (/book/web_development/9781785287756/6/ch06lvl1sec57/styling-based-on-feature-at