玩地图的选项

创建地图时可视化数据，您需要考虑一些重要的事情：要使用的投影，可用的缩放级别，图层请求要使用的默认图块大小，等等。这些重要部分大部分都包含在地图的属性中。

此食谱向您展示如何设置一些常见的地图属性。您可以在中找到此食谱的源代码 `ch01/ch01-map-options/` 。

做好准备

当您实例化一个在新 `ol.Map` 实例中，您可以选择将所有属性作为对象文字传递-这就是我们在第一个配方中所做的。在下一个食谱中，您将了解通过使用setter方法获得相似结果的另一种方式。

怎么做...

1. 就像我们在第一个配方中所做的一样，创建一个HTML页面来容纳地图，包括OpenLayers依赖项，并添加我们的自定义CSS和JavaScript文件。这次，将以下CSS放入自定义样式表中：

复制

```
.map {
  position: absolute;
  top: 0;
  bottom: 0;
  left: 0;
  right: 0;
}
.ol-mouse-position {
  top: inherit;
  bottom: 8px;
  left: 8px;
  background-color: rgba(255,255,255,0.4);
  border-radius: 2px;
  width: 100px;
  text-align: center;
  font-family: Arial, sans-serif;
  font-size: 12px;
}
```

2. 将以下内容放入您的自定义JavaScript文件中：

复制

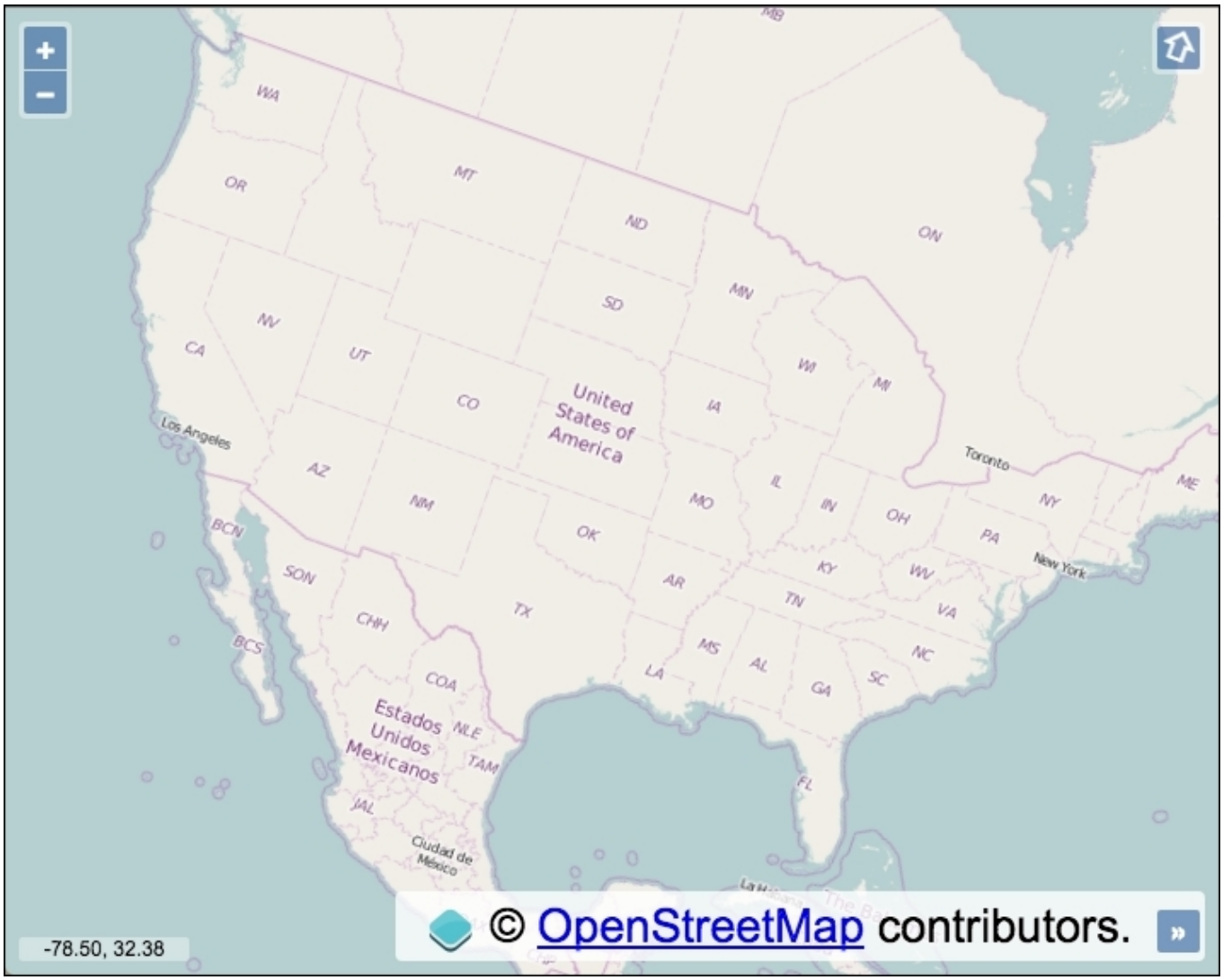
```
var map = new ol.Map({
  layers: [
    new ol.layer.Tile({
      source: new ol.source.OSM()
    })
  ]
});

var mousePositionControl = new ol.control.MousePosition({
  coordinateFormat: ol.coordinate.createStringXY(2),
  projection: 'EPSG:4326'
});

map.addControl(mousePositionControl);
map.setTarget('js-map');

var view = new ol.View({
  zoom: 4,
  projection: 'EPSG:3857',
```

如果现在打开 在浏览器中保存文件，您将看到类似于以下屏幕截图的内容：



怎么运行的...

除了CSS创建 在全屏地图上，我们还添加了一些新的CSS规则，这些规则可为地图上的鼠标位置控件设置样式（左下角）。这演示了使用一些简单的CSS样式化地图控件的简便性。鼠标位置控件的默认类名称为 `.ol-mouse-position` ，我们用来覆盖默认CSS。

我们在此食谱中介绍了一些新方法和属性，因此让我们一起介绍一下JavaScript：

复制

```
var map = new ol.Map({
  layers: [
    new ol.layer.Tile({
      source: new ol.source.OSM()
    })
  ]
});
```

When instantiating a new instance of `ol.Map` , we've passed in only the `layers` property at this point and saved a reference to the map instance in a variable named `map` .

Copy

```
var mousePositionControl = new ol.control.MousePosition({
  coordinateFormat: ol.coordinate.createStringXY(2),
  projection: 'EPSG:4326'
});
```

There's quite a bit going on in this snippet of JavaScript that we haven't seen before. When instantiating this new mouse position control, we passed in an object containing some additional settings.

The `coordinateFormat` property allows us to alter how the coordinates are displayed. This property expects an `ol.CoordinateFormatType` function that can be used to format an `ol.coordinate` array to a string. In other words, the `ol.coordinate.createStringXY` function returns the expected function type and formats the coordinates into a string, which we see onscreen. We specify the number of digits to include after the decimal point to `2` . Coordinates can get rather long, and we're not concerned with the level of accuracy here!

Let's take a look at the next property, `projection` . This tells OpenLayers to display the coordinates in the `EPSG:4326` projection. However, the default map projection is `EPSG:3857` . Due to this difference, OpenLayers must transform the projection from one type to another behind the scenes. If you were to remove this property from the control, it'll inherit the default map projection and you'll be presented with very different looking coordinates (in the `EPSG:3857` projection).

The `EPSG:4326` and `EPSG:3857` projections are boxed up with OpenLayers as standard. When you start dealing with other worldwide projections, you'll need to manually include the projection conversions yourself. Don't worry because there's a library for exactly this purpose, and we'll cover this later in this book.

Copy

```
map.addControl(mousePositionControl);
```

然后，使用 `addControl` 方法将鼠标位置控件添加到地图实例。这隐式扩展了默认地图控件。

复制

```
map.setTarget('js-map');
```

我们使用地图设置器方法之一来添加 `target` 属性和值。

复制

```
var view = new ol.View({
  zoom: 4,
  projection: 'EPSG:3857',
  maxZoom: 6,
  minZoom: 3,
  rotation: 0.34 // 20 degrees
});
```

我们引入了一些新的视图属性与此实例的视图： `projection` ， `maxZoom` ， `minZoom` ，和 `rotation` 。

该 `projection` 选项用于设置地图视图用来从图层渲染数据的投影。的投影 `EPSG:3857` 实际上与默认投影匹配，它也是OpenStreetMap使用的投影（这很重要，因为您需要确保tile服务接受投影的类型）。我们已在此处明确设置了它，仅用于演示目的。

设置 `maxZoom` 和 `minZoom` 属性会创建受限的缩放范围。这意味着用户只能查看可用缩放级别的子集。在这种情况下，他们无法将变焦范围扩大到变焦级别之外 `3` ，也无法将变焦范围扩大到变焦级别 `6` 。

该 `rotation` 物业旋转按弧度指定量的地图。您会注意到，一旦设置了旋转度，OpenLayers就会自动向地图添加旋转度控件。在此示例中，它出现在右上角。如果您感到迷失方向，可以单击此按钮，它将为您将地图旋转重置为0。

复制

```
view.setCenter([-10800000, 4510000]);
```

由于我们将 `view` 实例存储在变量中，因此可以像添加 `map` 实例一样轻松地添加其他属性。在这里，我们使用setter方法 `view` 来设置地图的初始中心位置。

复制

```
map.setView(view);
```

最后，我们 `view` 使用另一个有用的地图方法将完成的实例添加到地图实例 `setView` 。

注意

对于 `EPSG:4326` 和以外的投影 `EPSG:3857` ，您需要将该 `Proj4js` 项目（http://proj4js.org (http://proj4js.org)）包含在您的Web应用程序中。本书稍后将对此进行讨论。

EPSG代码是命名和分类可用投影集的一种方法。全球Coordinate Systems网站（http://epsg.io/ (http://epsg.io/)）是查找有关它们的更多信息的好地方。

还有更多...

所述 `EPSG:4326` 投影也被称为 **WGS84**，以度为单位进行度量。该 `EPSG:3857` 投影也知道的 **球形墨卡托**，以米为单位的坐标。

来自Google地图或OpenStreetMap之类的来源的图像是特殊情况，其中图像金字塔以前是使用Spherical Mercator投影创建的 `EPSG:3857` 。这意味着您在请求图块时无法设置投影，因为它是隐式的。

如果将图层放置在地图视图使用的投影之外的其他投影中，则它将无法按预期工作。

注意

Google Maps和OpenStreetMap等服务已预先渲染了栅格化的图像或图块，这些图像或图块构成了整个世界。这样可以避免服务器按需渲染图像，这意味着可以及时处理更多请求。图像形成金字塔的拼贴图案，从而在最小比例下，图块（金字塔的顶部）越少，并且随着比例的增加，该区域（金字塔的底部）就越多。您可以找到很好的解释，也可以找到此模式背后的一些有趣历史此处：https (https://www.e-education.psu.edu/geog585/node/706) ://www.e-education.psu.edu/geog585/node/706 (https://www.e-education.psu.edu/geog585/node/706)。

也可以看看

- 该**管理图的栈层**食谱
- 该**管理地图的控制**食谱
- 在**与预测工作**配方在第7章 (/book/web_development/9781785287756/7)，**超越基础。**

上一节 (/book/web_development/9781785287756/1/ch01lvl1sec10/creating-a-simple-fullscreen

下一节 (/book/web_development/9781785287756/1/ch01lvl1sec12/managing-the-map-s-stack-