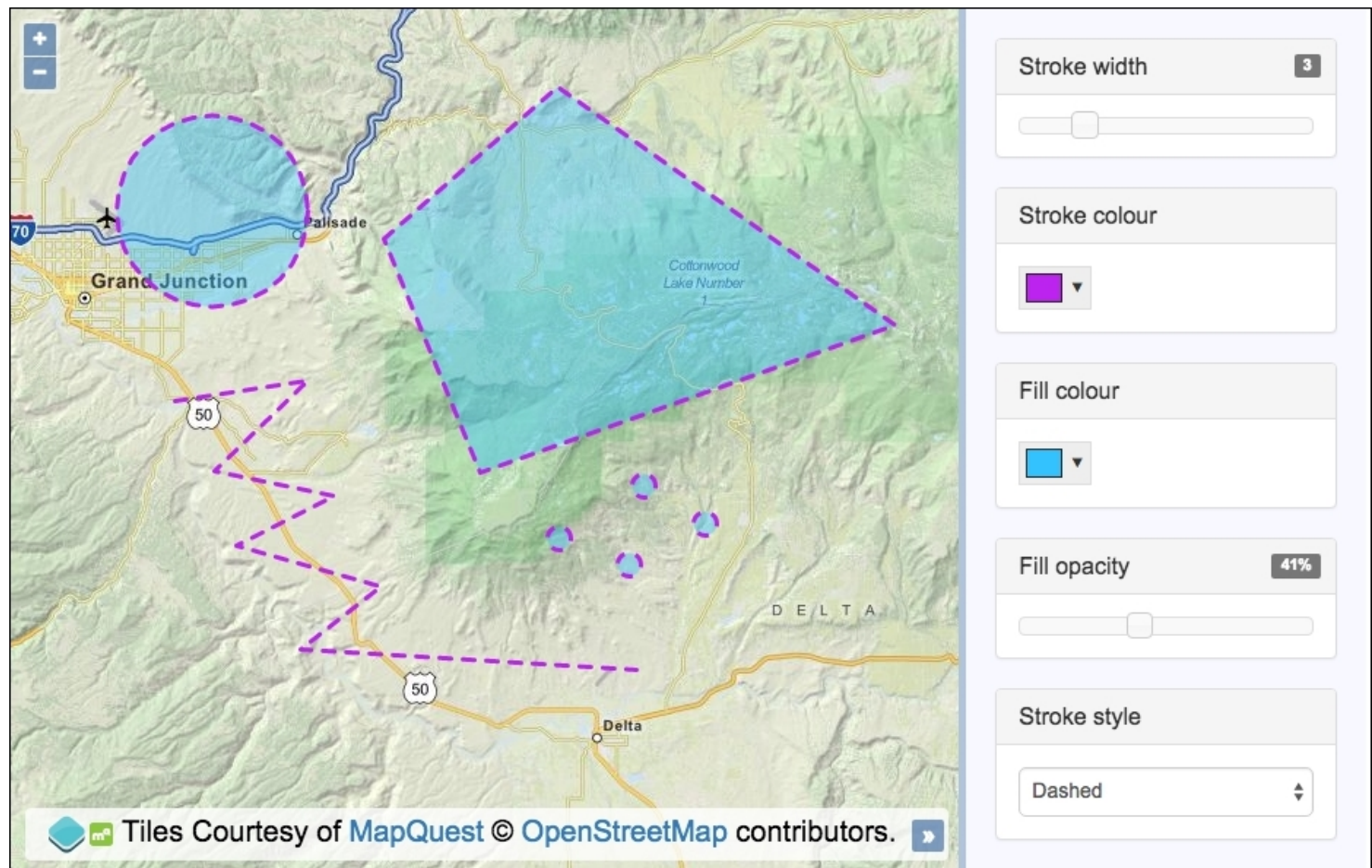


样式层

至 演示一些可用的要素样式选项，我们将在侧面板中创建一个实时编辑器，以对已经加载到地图上的要素进行样式设置。为了实现此目标，我们将使用jQuery UI库为颜色选择器小部件创建滑块和jQuery插件Spectrum。可以在中找到源代码 `ch06/ch06-styling-layers` 。我们最终得到的内容类似于以下屏幕截图：



对于本食谱，我们将在图层级别应用样式，以便图层中的所有要素都继承样式。所有可自定义样式的当前状态将反映在侧栏中。

我们将实例化属于该 `ol.style` 对象的类的许多实例，例如 `ol.style.Stroke` 。让我们看一下它是如何实现的。

做好准备

的 源代码有两个主要部分：一个用于放置所有控件的HTML，另一个用于JavaScript代码。

HTML部分包含许多标记，这些标记可通过JavaScript转换为交互式小部件，例如 滑块和颜色选择器。正如前面提到的，我们还使用了jQuery的，jQuery用户界面 (<https://jqueryui.com> (<https://jqueryui.com>)) 和Spectrum (<https://bgrins.github.io/spectrum> (<https://bgrins.github.io/spectrum>)) 库。由于它们不是本食谱的目标，因此HTML不会在“ 如何做...”部分列出。我鼓励您看一下本书代码捆绑中的代码。

怎么做...

为了创建自己的自定义图层样式，请遵循以下说明：

- 1 创建包括OpenLayers依赖项和其他库依赖项的HTML文件（请参阅HTML代码的“ **准备就绪**”部分）之后，创建一个自定义JavaScript文件并实例化一个新 `map` 实例，如下所示：

复制

```
var map = new ol.Map({
  view: new ol.View({
    zoom: 10, center: [-12036691, 4697972]
  }),
  target: 'js-map',
  layers: [
    new ol.layer.Tile({
      source: new ol.source.MapQuest({layer: 'osm'})
    })
  ]
});
```

- 2 为 `source` 要加载到功能的GeoJSON文件中的实例创建矢量层：

复制

```
var vectorLayer = new ol.layer.Vector({
  source: new ol.source.Vector({
    url: 'features.geojson',
    format: new ol.format.GeoJSON({
      defaultDataProjection: 'EPSG:3857'
    })
  })
});
```

- 3 设置将样式放在一起的函数，以便可以用来设置完整的矢量层样式，如下所示：

复制

```
var setStyles = function() {
    vectorLayer.setStyle(new ol.style.Style({
        stroke: strokeStyle(),
        fill: fillStyle(),
        image: new ol.style.Circle({
            fill: fillStyle(),
            stroke: strokeStyle(),
            radius: 8
        })
    }));
};
```

- 4 组起来 `stroke-width` , 并 `fill-opacity` 与jQuery UI的滑块。该操作的处理程序 `slide` 调用我们的 `setStyles` 方法, 它们在DOM中显示相关的值供用户查看, 如下所示:

```
$('#js-stroke-width').slider({
    min: 1, max: 10, step: 1, value: 1,
    slide: function(event, ui) {
        $('#js-stroke-width-value').text(ui.value);
        setStyles();
    }
});
$('#js-fill-opacity').slider({
    min: 0, max: 100, step: 1, value: 50,
    slide: function(event, ui) {
        $('#js-fill-opacity-value').text(ui.value + '%');
        setStyles();
    }
});
```

复制

- 5 借助Spectrum jQuery插件, 将 `stroke` 和 `fill` 颜色的标记转换为交互式颜色选择器。选择新颜色后, 调用我们的 `setStyles` 方法:

```
$('#js-stroke-color, #js-fill-color').spectrum({
    color: 'black', change: setStyles
});
```

复制

- 6 在描边样式 `select` 菜单中侦听更改, 然后调用我们的 `setStyles` 方法:

```
$('#js-stroke-style').on('change', setStyles);
```

复制

- 7 设置返回填充样式的函数, 如下所示:

复制

```
var fillStyle = function() {
    var rgb = $('#js-fill-color').spectrum('get').toRgb();
    return new ol.style.Fill({
        color: [
            rgb.r, rgb.g, rgb.b,
            $('#js-fill-opacity').slider('value') / 100
        ]
    });
};
```

8 组返回返回笔触样式的函数，如下所示：

复制

```
var strokeStyle = function() {
    return new ol.style.Stroke({
        color: $('#js-stroke-color').spectrum('get').toHexString(),
        width: $('#js-stroke-width').slider('value'),
        lineDash: $('#js-stroke-style').val() === 'solid'
            ? undefined : [8]
    });
};
```

9 通过调用 `setStyles` 函数结束并将向量层添加到 `map` 实例，如下所示：

复制

```
setStyles();
map.addLayer(vectorLayer);
```

怎么运行的...

如果我们逐行介绍的话，这里有很多JavaScript。相反，我们要做的是充分利用关键功能，使您可以自己理解整体代码：

复制

```
var setStyles = function() {
    vectorLayer.setStyle(new ol.style.Style({
        stroke: strokeStyle(),
        fill: fillStyle(),
        image: new ol.style.Circle({
            fill: fillStyle(),
            stroke: strokeStyle(),
            radius: 8
        })
    }));
};
```

我们创建了一个函数，`setStyles` 当用户在侧面板中自定义样式时会在代码中的多个位置调用该函数。调用此方法时，它将返回 `ol.style.Style` 对象的完整实例，并将其传递给矢量层的 `setStyle` 方法。反过来，这导致使用最新样式配置重新渲染地图上的要素。

为的笔画属性 `ol.style.Style` 分配了由该 `strokeStyle` 方法生成的笔画样式的值, 并且该 `fill` 属性由该 `fillStyle` 方法生成。

该 `image` 属性是 `ol.style.Circle` 样式对象的实例, 即几何显示点几何的类型。我们重复使用 `fill` 和 `stroke` 方法, 以提供 `fill` 和 `stroke` 属性的值。我们为圆样式设置了静态 `radius` 值 `8` 。

复制

```
$('#js-stroke-width').slider({
  min: 1, max: 10, step: 1, value: 1,
  slide: function(event, ui) {
    $('#js-stroke-width-value').text(ui.value);
    setStyles();
  }
});
```

让我们看一下 `slider` 笔触宽度的jQuery设置之一。熟悉此 `slider` 功能后, 您将可以确定不透明度的其他 `slider` 功能如何 `fill` 适合本配方的实现。

我们将宽度限制为介于 `1` 和之间 `10` (包括 `min` 和 `max` 属性) (包括和属性), 默认宽度为 `1`。我们为 `slide` 在 `($('#js-stroke-width-value').text(ui.value))` 边栏中显示更新值的操作注册处理程序; 尚不影响实际特征相对于地图的样式。

然后 `setStyles`, 我们调用方法, 因为我们希望最新的样式立即反映在地图上。

复制

```
$('#js-stroke-color, #js-fill-color').spectrum({
  color: 'black', change: setStyles
});
```

我们将在本章中介绍的其他小部件类型是颜色选择器 (我们将不介绍笔触样式 `select` 菜单的处理程序, 因为它应该很容易理解)。

在jQuery选择器中, 我们选择了和的两个 `color` 输入元素, `fill` 并 `stroke` 传入了默认颜色, `black` 并选择了一个 `setStyles` 在选择新颜色时调用我们方法的处理程序。

复制

```
var fillStyle = function() {
  var rgb = $('#js-fill-color').spectrum('get').toRgb();
  return new ol.style.Fill({
    color: [
      rgb.r, rgb.g, rgb.b,
      $('#js-fill-opacity').slider('value') / 100
    ]
  });
};
```

当我们从DOM元素初始化小部件时以及用户每次对其进行更新时，Spectrum都会保留填充颜色（来自颜色选择器）。为了动态地获取最新的颜色，我们访问相同的DOM元素，并调用 `get` 方法，然后调用该 `toRgb` 方法-两者均来自Spectrum，并将结果存储在 `rgb` 变量中。结果是将红色值分配给 `r` 属性的对象，绿色和蓝色值依此类推。

我们两个实例化并返回的实例 `ol.style.Fill`，该实例接受的单个对象属性 `color`。该值可以是OpenLayer类型的数组 `ol.Color`（[红色，绿色，蓝色，alpha]）。我们使用RGB值，然后是alpha值来检索并填充数组。

alpha值取自jQuery UI滑块DOM元素（`slider('value')`）。该数字转换为浮点数。例如55；在这种情况下，我们将55除以 `100`（0.55）。这很重要，因为 `ol.Color` 类型数组期望alpha为从0-1（含0）开始的浮点值。

如本食谱前面所述，我们的 `setStyles` 方法使用该 `fillStyle` 方法填充其某些属性。

复制

```
var strokeStyle = function() {
  return new ol.style.Stroke({
    color: $('#js-stroke-color').spectrum('get').toHexString(),
    width: $('#js-stroke-width').slider('value'),
    lineDash: $('#js-stroke-style').val() === 'solid'
      ? undefined : [8]
  });
};
```

与我们的 `fillStyle` 函数类似，笔画样式构造函数（`ol.style.Stroke`）的实例会同时返回并实例化。更多特性可用于中风的造型，从中我们填充 `color`，`width` 以及 `lineDash` 与值的属性。

对于该 `color` 属性，我们再次利用Spectrum来检索最新颜色选择器DOM元素的值，但是这次，我们将其传递给 `toHexString` 库方法，以字符串十六进制格式生成颜色。的 `color` 属性 `ol.style.Stroke` 也可以接受 `ol.Color` 数组。

对于 `width` 属性，我们只需要直接从jQuery `slider` DOM元素中获取值即可。

该 `lineDash` 属性的值是根据用户的选择推断出来的。如果用户从菜单中选择“**无**”（带有 `option value` 的 `solid`）`select`，那么我们传入 `undefined`，这意味着该 `stroke` 实例不需要破折号。否则，我们传入一个带有单个数字条目的数组 `8`。

当您传递数字数组时，您将定义一个虚线模式。数字代表交替的虚线和间隙的长度。因此，`8` 就象我们在这里所做的那样，它本身等于[8, 8]（假定间隙与短划线的长度相同）。

您可以自行设计更复杂的模式。我鼓励您尝试其他选择。Mozilla有关于笔划破折号数组的有用指南在<https://developer.mozilla.org/en/docs/Web/SVG/Attribute/stroke-dasharray> (https://developer.mozilla.org/en/docs/Web/SVG/Attribute/stroke-dasharray)。

该 `ol.style.Stroke` 构造函数接受其他属性，如 `lineCap` ，和 `lineJoin` ，你可能会觉得有趣。

还有更多...

这里可能出现的问题是，在样式设置，应用于矢量层的规则或直接应用于单个要素的样式时，什么优先？

答案是样式从下到上。这意味着，如果我们直接在要素上指定了样式，则将使用该样式进行渲染。否则，分配给矢量层的任何样式都将应用于其要素。

也可以看看

➤ **基于要素属性的样式配方**

➤ **该造型交互渲染意图食谱**

◀ [上一节 \(/book/web_development/9781785287756/6/ch06lvl1sec54/introduction\)](/book/web_development/9781785287756/6/ch06lvl1sec54/introduction)

[下一节 ▶ \(/book/web_development/9781785287756/6/ch06lvl1sec56/styling-features-based-on-g](/book/web_development/9781785287756/6/ch06lvl1sec56/styling-features-based-on-g)

