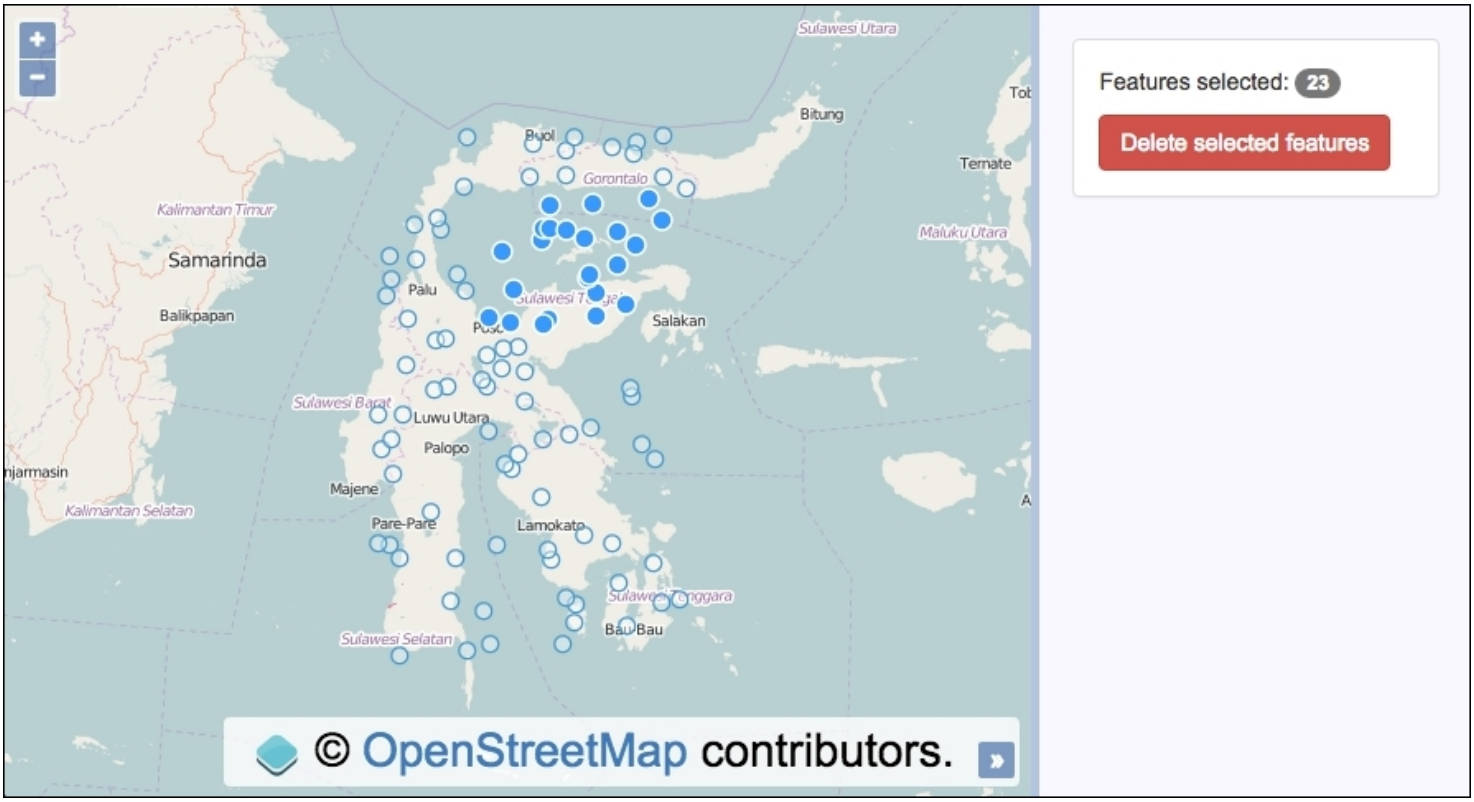
通过拖出选择区域来选择特征

一种常见 在矢量层中使用要素时的动作是它的选择，当然，OpenLayers具有一些可供我们使用的要素选择控件。我们已经看到了这一点表现在早期的食谱，如**删除或利用覆盖克隆功能**，在配方第3章(/book/web_development/9781785287756/3)，**工作与矢量图层**。此食谱向我们展示了如何通过单击或点击手势一次选择一个功能。

本食谱将向您展示如何使用交互方式一次选择多个功能。的 `ol.interaction.Select` 类是呈现意图和选择的特征进行分组，并且另一个是 `ol.interaction.DragBox` ，它是用来使用户能够拖拽出在地图上的矩形。这两个控件的共同作用将产生多选功能。

选中这些功能后，我们将显示所选计数，并使用户能够从侧边栏中删除所选功能。

来源 可以在中找到代码 `ch07/ch07-dragbox-selection` ，其外观类似于以下屏幕截图：



事不宜迟，让我们将所有功能放在一起。

怎么做...

1 创建一个HTML文件，并添加OpenLayers库依赖项和一个 `div` 元素。特别是，请确保您具有以下标记，该标记将用于显示选定的功能计数以及删除按钮：

复制

```
<p>Features selected: <span id="js-selected">0</span></p>
<button id="js-delete" disabled="disabled">Delete selected features</button>
```

2 创建一个 自定义JavaScript文件，并 `map` 使用 `view` 实例和栅格图层实例化实例：

复制

```
var map = new ol.Map({
  view: new ol.View({
    zoom: 6, minZoom: 6, maxZoom: 6, center: [13484714, -266612]
  }), target: 'js-map',
  layers: [
    new ol.layer.Tile({source: new ol.source.OSM()})
  ]
});
```

3 接下来，设置一个带有获取本地GeoJSON文件并将其添加到 `map` 实例的源的矢量层：

复制

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

4 缓存一些可能经常访问的DOM元素，如下所示：

复制

```
var selectedCount = document.getElementById('js-selected');
var deleteButton = document.getElementById('js-delete');
```

5 实例化两个交互的实例，并将它们添加到 `map` 实例中：

复制

```
var select = new ol.interaction.Select();
map.addInteraction(select);
var dragbox = new ol.interaction.DragBox();
map.addInteraction(dragbox);
```

6 创建一个可重用的函数，该函数将重置一些UI状态并清除功能选择：

复制

```
var reset = function() {
  select.getFeatures().clear();
  selectedCount.innerHTML = 0;
  deleteButton.setAttribute('disabled', 'disabled');
};
```

7 订阅 `DragBox` 交互中的一些事件，并在有以下功能选择时更新UI：

复制

```
dragbox.on('boxstart', reset);
dragbox.on('boxend', function() {
  var extent = dragbox.getGeometry().getExtent();
  var count = 0;
  vectorLayer.getSource()
  .forEachFeatureIntersectingExtent(extent, function(feature) {
    select.getFeatures().push(feature);
    count++;
  });
  selectedCount.innerHTML = count;

  if(count > 0) {
    deleteButton.removeAttribute('disabled');
  } else {
    deleteButton.setAttribute('disabled', 'disabled');
  }
});
```

8 订阅 `click` 删除按钮元素上的事件并删除已选择的所有功能：

复制

```
deleteButton.addEventListener('click', function() {
  select.getFeatures().forEach(function(feature) {
    vectorLayer.getSource().removeFeature(feature);
  });
  reset();
});
```

怎么运行的...

我们已经使用Bootstrap CSS框架来处理此食谱的样式，但是已省略了许多HTML和所有CSS。请查看随附的源代码以获取完整的实现。

让我们将注意力转向该代码的新引入的概念：

复制

```
var select = new ol.interaction.Select();
map.addInteraction(select);
var dragbox = new ol.interaction.DragBox();
map.addInteraction(dragbox);
```

我们已经创建了两个交互作用，可以统一执行它们的职责。它们都是 `ol.interaction` 该类的子类。请记住，OpenLayers中的交互并不带有放置在地图上的物理DOM元素。

复制

```
var reset = function() {
  select.getFeatures().clear();
  selectedCount.innerHTML = 0;
  deleteButton.setAttribute('disabled', 'disabled');
};
```

当此 `reset` 功能是调用后， `select` 将检索（ `getFeatures` ）存储在select交互（）中的所有功能，然后通过该 `clear` 方法将其从 `ol.Collection` 数组中丢弃。

然后更新UI，以将功能计数恢复为零并禁用删除按钮（因为未选择任何功能）。用户界面将反映出用户可以再次选择更多功能。在代码执行流的上下文中查看此功能时，它会更有意义。

复制

```
dragbox.on('boxstart', reset);
```

当用户开始在地图上拖动框时，交互将发布 `boxstart` 事件。我们订阅此事件，并 `reset` 通过将 `reset` 函数分配为处理程序来确保UI返回到状态。

复制

```
dragbox.on('boxend', function() {
  var extent = dragbox.getGeometry().getExtent();
  var count = 0;
  vectorLayer.getSource()
  .forEachFeatureIntersectingExtent(extent, function(feature) {
    select.getFeatures().push(feature);
    count++;
  });
  selectedCount.innerHTML = count;
```

这是 `boxend` 事件订阅处理程序的前半部分。用户释放鼠标按钮时，将发布此事件。我们从交互实例中检索最后绘制的框的几何形状，并获得范围（ `dragbox.getGeometry().getExtent()` ）。这特别有用，因为它允许我们针对向量源构造查询。

我们将看到在拖出的矩形的边界框内有多少要素，因此将计数设置为零。

向量源有一个称为 `forEachFeatureIntersectingExtent` （很容易说明）的方法来查询源，并返回与范围相交的所有特征。还有一个类似的可用方法 `forEachFeatureInExtent` ，其行为略有不同，即如果要素的边界框在范围内，则返回true，而不是要素的几何图形实际上是否与范围相交。您可以在上阅读有关差异的更多信息OpenLayers文档（http://openlayers.org/ (http://openlayers.org/)）。

该 `forEachFeatureIntersectingExtent` 方法期望第一个参数是 `extent` （我们拖动框的范围），第二个参数是迭代器函数，即将 `feature` in迭代作为第一个参数传递。我们将 `feature` 放入选择交互（ `select.getFeatures().push(feature)` ）的选择数组中。这样， `select` 交互将跟踪选定的功能，并使用适当的渲染意图对它们进行样式设置。

由于 `count` 变量在每次循环（ `count++` ）时都会增加，因此我们将选定特征的最终总和注入DOM中以进行显示。

复制

```
  if(count > 0) {
    deleteButton.removeAttribute('disabled');
  } else {
    deleteButton.setAttribute('disabled', 'disabled');
  }
```

事件处理程序的最后一半检查是否 `count` 超过 `0` 。如果是这样，则表明功能选择成功，因此可以通过 `disabled` 从HTML删除属性来启用删除按钮。如果用户愿意，他们可以否定从矢量源中删除特征。

相反，如果计数不超过 `0` ，则我们确保 `disabled` 已设置按钮上的属性（因为用户无法删除任何内容）。

复制

```
deleteButton.addEventListener('click', function() {
  select.getFeatures().forEach(function(feature) {
    vectorLayer.getSource().removeFeature(feature);
  });
  reset();
});
```

最后，我们 `click` 在按钮上订阅该事件。当单击时，我们获取 `select` 控件中保留的功能，然后 `feature` 在集合中的每个功能上循环。对于每个 `feature` ，我们检索方法中的向量源和链 `removeFeature` ，并在 `feature` 迭代过程中传递。指示矢量源层删除 `feature` 。

使用循环和删除功能后，我们将UI更新为重置状态以反映所做的更改，然后准备进行下一个选择。

还有更多...

您可能会注意到 `DragBox` 互动只要在地图上单击即可激活。对于大多数应用程序，这可能是不受欢迎的默认行为。要从UI元素（例如，按钮）触发交互的激活或取消激活，可以调用 `setActive` 交互方法并传入 `true` / `false` （或从地图中删除或添加交互）以切换交互。

或者，您可以决定仅在结合 `click` 手势按下键盘上的特定键时才启用拖动。例如，如以下交互所示：

复制

```
var dragbox = new ol.interaction.DragBox({
  condition: ol.events.condition.shiftKeyOnly
});
```

的 `DragBox` 只有在**按住Shift键并按住某个** `click` 事件的同时，才激活先前的交互。

也可以看看

- 在**创建自定义控制**配方
- 第6章 (/book/web_development/9781785287756/6)，**样式化功能中**的**样式化交互呈现意图**配方 (/book/web_development/9781785287756/6)

上一节 (/book/web_development/9781785287756/7/ch07lvl1sec62/creating-a-custom-control)

下一节 (/book/web_development/9781785287756/7/ch07lvl1sec64/transitioning-between-wea

---