

## 在地图视图中移动

除非您要创建一个完全静态的地图，而无需用户平移，缩放或旋转所需的控件，您希望用户能够导航和浏览地图。

在某些情况下，内置控件不够用。想象一下一个Web应用程序，用户可以在其中搜索术语，例如“**Everest**”，并且该应用程序必须找到其位置并平移到该位置。在这种情况下，您需要按代码导航而不使用控件。

此食谱向您展示了一些编程方式，无需使用默认控件即可在地图上移动。可以在中找到源代码

`ch01/ch01-moving-around`，这是我们的最终结果：



该应用程序包含一个选择欧洲城市，更改后会将地图平移到所选城市。当前的缩放，旋转，经度和纬度值通过地图交互保持最新。还可以手动编辑这些输入字段以更新其各自的地图属性。



注意

我们已经省略了创建应用程序布局所必需的完整HTML和CSS代码；因此，如果您对完整的代码感兴趣，可以查看Packt Publishing网站上提供的源代码。

## 怎么做...

- 1 创建具有OpenLayers依赖项的HTML文件。大多数HTML都是自解释性的，但是特别是以下是城市选择菜单的HTML（这将有助于我们以后理解JavaScript）：

复制

```
<select id="js-city">
  <option value="12.5,41.9">Rome (Italy)</option>
  <option value="30.517,50.45">Kiev (Ukraine)</option>
  <option value="-9.183,38.7">Lisbon (Portugal)</option>
  <option value="-0.117,51.5">London (England)</option>
  <option value="14.417,50.083">Prague (Czech Rep)</option>
</select>
```

- 2 创建一个 `map` 实例如下：

复制

```
var map = new ol.Map({
  layers: [
    new ol.layer.Tile({
      source: new ol.source.Stamen({
        layer: 'watercolor'
      })
    })
  ],
  target: 'js-map',
  view: new ol.View({
    zoom: 6,
    center: ol.proj.fromLonLat([12.5, 41.9])
  })
});
```

- 3 将某些DOM元素缓存到可重用的变量中：

复制

```
var citySelect = document.getElementById('js-city');
var zoomInput = document.getElementById('js-zoom');
var rotateInput = document.getElementById('js-rotate');
var lonInput = document.getElementById('js-lon');
var latInput = document.getElementById('js-lat');
```

- 4 向事件视图添加一些事件侦听器以及事件处理函数：

复制

```

var updateUI = function(event) {
    var view = event && event.currentTarget || map.getView();
    zoomInput.value = view.getZoom();
    rotateInput.value = view.getRotation();

    var centerLonLat = ol.proj.toLonLat(view.getCenter());
    lonInput.value = centerLonLat[0].toFixed(3);
    latInput.value = centerLonLat[1].toFixed(3);
};
updateUI();

map.getView().on([
    'change:center',
    'change:resolution',
    'change:rotation'
], updateUI);

```

## 5 创建一个助手 设置新地图视图中心的功能：

复制

```

var setCenter = function(lon, lat) {
    map.getView().setCenter(ol.proj.fromLonLat([
        parseFloat(lon), parseFloat(lat)
    ]));
};

```

## 6 创建事件侦听器和处理程序以更新输入字段：

复制

```

window.addEventListener('keyup', function(event) {
    switch(event.target.id) {
        case 'js-zoom':
            map.beforeRender(ol.animation.zoom({
                resolution: map.getView().getResolution(),
                duration: 150
            }));
            map.getView().setZoom(parseInt(event.target.value, 10));
            break;

        case 'js-rotate':
            map.beforeRender(ol.animation.rotate({
                rotation: map.getView().getRotation(),
                duration: 250
            }));
            map.getView().setRotation(parseFloat(event.target.value));
            break;

        case 'js-lon':

```

## 7 为城市选择创建事件侦听器和处理程序：

复制

```
citySelect.addEventListener('change', function() {
  map.beforeRender(ol.animation.pan({
    source: map.getView().getCenter(),
    duration: 500
  }));
  setCenter.apply(null, this.value.split(','));
});
```

## 怎么运行的...

还有一点在此，我们介绍了对多种地图导航方法的手动控制。我们还迷上了地图事件，动画和投影转换。现在是时候仔细看看发生了什么：

复制

```
new ol.layer.Tile({
  source: new ol.source.Stamen({
    layer: 'watercolor'
  })
})
```

此配方的平铺服务来自**雄蕊**源，带有水彩层样式。这是OpenLayers内置支持并易于包含的另一个来源。

复制

```
view: new ol.View({
  zoom: 6,
  center: ol.proj.fromLonLat([12.5, 41.9])
})
```

对于此食谱，我们使用经度和纬度值在地图上导航。但是，地图视图的默认投影为 `EPSG:3857`（球形墨卡托），并且 `EPSG:4326` 投影中的经度和纬度。我们需要一种转换这些经度和纬度坐标的方法。

幸运的是，我们 `ol.proj` 有许多有用的方法，其中一种是将坐标从经度和纬度转换为

`EPSG:3857`，我们刚刚使用过。您还可以将目标投影作为第二个参数传递给 `fromLonLat`，但是默认的目标投影仍然是 `EPSG:3857`，因此我们无需理会。

复制

```
var citySelect = document.getElementById('js-city');
var zoomInput = document.getElementById('js-zoom');
var rotateInput = document.getElementById('js-rotate');
var lonInput = document.getElementById('js-lon');
var latInput = document.getElementById('js-lat');
```

用户进行交互的DOM元素已被缓存到变量中以提高效率。我们引用这些元素是为了检索和更新值。

复制

```
var updateUI = function(event) {
  var view = event && event.currentTarget || map.getView();
  zoomInput.value = view.getZoom();
  rotateInput.value = view.getRotation();

  var centerLonLat = ol.proj.toLonLat(view.getCenter());
  lonInput.value = centerLonLat[0].toFixed(3);
  latInput.value = centerLonLat[1].toFixed(3);
};
updateUI();
```

功能 `updateUI` 已创建被调用以便将输入字段与当前地图状态同步。此函数将在页面初始化时被调用或作为事件处理程序。为了解决这两种情况，地图视图将从事件参数（如果有）中派生（`event.currentTarget` 在这种情况下将是地图视图），或者我们自己抓住它（`map.getView()`）。当然，我们可以 `map.getView` 在两种情况下都使用过，但是最好熟悉一些可用的地图事件属性。

`get` 使用视图（`getZoom` 和 `getRotation`）提供的简单方法，可以轻松更新缩放和旋转值。

中心位置需要做更多的工作。请记住，地图视图的投影位于中 `EPSG:3857`，但是我们要以经度和纬度显示坐标。使用 `ol.proj.toLonLat` 将坐标从 Spherical Mercator 转换为的方法设置视图时，我们做的与之前相反 `EPSG:4326`。此方法接受第二个参数来标识源投影。默认的源投影为 `EPSG:3857`，无论如何都与我们的地图视图投影匹配，因此我们可以跳过指定。

结果返回一个数组，该数组存储在中 `centerLonLat`。然后，我们检索要在输入字段中显示的各个值，并将小数点约束为 3。

[复制](#)

```
map.getView().on([
  'change:center',
  'change:resolution',
  'change:rotation'
], updateUI);
```

本 `ol.View` 类有一个 `on` 使我们能够从视图订阅特定的事件，并指定一个事件处理方法。我们重视三个事件监听器 `view`：`center`，`resolution`，和 `rotation`。分辨率事件侦听器用于缩放级别的更改。这些视图属性中的任何一个发生更改时，`updateUI` 都会调用我们的事件处理程序。

[复制](#)

```
var setCenter = function(lon, lat) {
  map.getView().setCenter(ol.proj.fromLonLat([
    parseFloat(lon), parseFloat(lat)
  ]));
};
```



在这个食谱中，我们需要在代码中的多个不同位置设置一个新的中心位置。为了使自己更轻松，我们创建了一个 `setCenter` 函数，该函数使用 `lon` 和 `lat` 值。它将提供的经度和纬度坐标转换为地图投影坐标，并设置新的中心位置。

由于经度和纬度值将作为字符串来自输入元素，因此我们将这些值传递到 `parseFloat` JavaScript方法中，以确保它们处于OpenLayers所需的类型格式中。

复制

```
window.addEventListener('keyup', function(event) {  
  switch(event.target.id) {
```

我们将全局 `keyup` 事件侦听器附加到window对象，而不是为每个输入字段添加单个事件侦听器。调用此事件处理程序时，我们通过通过检查 `switch` 语句检查目标元素ID属性来确定要执行的操作。

例如，如果修改了缩放输入字段值，那么目标ID将是 `js-zoom` 因为HTML标记为 `<input type="number" id="js-zoom">` ：

复制

```
case 'js-zoom':  
  map.beforeRender(ol.animation.zoom({  
    resolution: map.getView().getResolution(),  
    duration: 150  
  }));  
  map.getView().setZoom(parseInt(event.target.value, 10));  
  break;
```

第一个开关盒用于缩放输入字段。与其在地图视图上简单地设置新的缩放级别，不如为缩放级别之间的过渡设置动画。为此，我们在通过 `ol.Map.beforeRender` 方法渲染缩放更改之前添加了要调用的函数。它期望类型为的一个或多个函数 `ol.PreRenderFunction` ， `ol.animation.zoom` 方法返回此特定的函数类型，从而为分辨率转换设置动画。

的 `resolution` 属性 `ol.animation.zoom` 提供了动画的起点，即当前的分辨率。该 `duration` 属性以毫秒为单位，因此这将是一个快速而生动的动画。

附加prerender函数后，我们将获取用户输入值并 `setZoom` 通过 `parseInt` JavaScript方法设置最终缩放级别（），以确保将输入字段字符串转换为OpenLayers的预期数字类型。

复制

```
case 'js-rotate':  
  map.beforeRender(ol.animation.rotate({  
    rotation: map.getView().getRotation(),  
    duration: 250  
  }));  
  map.getView().setRotation(parseFloat(event.target.value));  
  break;
```



这个开关盒捕获旋转输入字段。与以前的缩放控件类似，我们希望再次为过渡设置动画。为此，我们使用创建了一个`prerender`函数 `ol.animate.rotate` 。我们传入视图的当前旋转以及自定义的 `250` 毫秒持续时间。此后，我们使用 `setRotation` 地图视图方法根据输入字段值设置新的旋转量。再次，我们确保通过该 `parseFloat` 方法将输入字符串转换为OpenLayers的float值。

复制

```
case 'js-lon':
    setCenter(event.target.value, latInput.value);
break;

case 'js-lat':
    setCenter(lonInput.value, event.target.value);
break;
```

这些开关盒匹配经度和纬度输入字段的变化。随着经度和纬度的变化，我们决定将其捕捉到新的中心位置，而不是为其设置动画。我们调用我们自己的 `setCenter` 方法，该方法先前已经讨论过，可以使用经度和纬度值。当经度和纬度值配对时，未更改的值将从相应的输入字段中获取。

复制

```
citySelect.addEventListener('change', function() {
    map.beforeRender(ol.animation.pan({
        source: map.getView().getCenter(),
        duration: 500
    }));
    setCenter.apply(null, this.value.split(','));
});
```

最后，我们将一个 `change` 事件附加到城市选择菜单。我们已决定将平移动画从原来的中心位置移动到新的中心位置。就像缩放和旋转过渡一样，我们使用特定于平移的 `ol.animation.pan` 方法。我们为 `source` 属性提供起始位置，并设置半秒的持续时间。

一旦预渲染功能到位，我们就可以设置新的中心位置。再次，我们调用自定义 `setCenter` 函数为我们执行此操作。

的HTML 城市选择菜单中的特定选项包含经度和纬度值作为字符串。例如，如果要平移到伦敦，则选项内的值是逗号分隔的 `string: <option value="-0.117,51.5">London (England)</option>` 。我们 `"-0.117,51.5"` 使用JavaScript `split` 方法将此字符串 ( ) 转换为数组，以提供不同的值分隔。但是，我们的 `setCenter` 函数需要两个参数，而不是值的数组。为了解决这个问题，我们使用JavaScript `apply` 方法，该方法使用 `setCenter` 参数数组进行调用，产生相同的结果。

这样就彻底了解了如何在没有默认控件的情况下浏览地图，从而提供了极大的灵活性。

也可以看看

---

◀ 上一节 (/book/web\_development/9781785287756/1/ch01lvl1sec13/managing-the-map-s-control)

下一节 ▶ (/book/web\_development/9781785287756/1/ch01lvl1sec15/restricting-the-map-s-extension)

---

