利用地图固定链接

一个伟大的功能 浏览万维网时，可以为您感兴趣的网站添加书签。这些书签为您提供了一种保存链接并在以后返回到相同内容的方法。

某些网页在URL中包含变量，这些变量标识您正在访问的特定内容。您可以与其他人共享URL的副本，他们也可以完全看到您看到的内容。当然，我们也可以将这种便利应用于网络映射应用程序。

如果我们可以共享地图的URL（永久链接），以便在将其加载到其他人的设备上时可以恢复地图上的相同位置，那不是很好吗？我们可以使用JavaScript代码在客户端上实现此目的，而无需在服务器端使用逻辑来帮助构建基于URL的地图加载。

为了说明这一点，我们将提供一个带有键和值的哈希值的URL（JavaScript应用程序中的一种常见技术）。这些值将告诉我们地图视口的中心，缩放级别以及可用的地图控件。

可以在中找到源代码 `ch07/ch07-map-permalinks` ，这是基于URL内容的实例化地图的屏幕截图：

怎么做...

1. 创建一个具有OpenLayers依赖项的HTML文件以及一个 `div` 用于保存地图的元素。
2. 创建一个 自定义JavaScript文件并设置正则表达式以与URL以及其他一些默认变量赋值进行匹配：

复制

```
var viewHashRegex = /view=([^z]+)/;
var controlHashRegex = /controls=\[(.+)\]/;
var defaultView = [-8726204, 4937946, 12];
var view, viewArray, controls, controlsArray;
```

3. 实例化 `map` 不 `controls` 带栅格图块层并带有栅格图块层的新实例：

复制

```
var map = new ol.Map({
  target: 'js-map',
  layers: [
    new ol.layer.Tile({
      source: new ol.source.Stamen({layer: 'terrain'})
    })
  ], controls: []
});
```

4 检查URL哈希内容的可用性，并检索 `view` 和 `controls` 细节（如果存在）：

复制

```
if (window.location.hash) {
  view = window.location.hash.match(viewHashRegex);
  controls = window.location.hash.match(controlHashRegex);

  if (view) {
    viewArray = view[1].split(',');

    if (viewArray.length === 3) {
      defaultView[0] = viewArray[0];
      defaultView[1] = viewArray[1];
      defaultView[2] = viewArray[2];
    }
  }

  if (controls) {
    controlsArray = controls[1].split(',');
    controlsArray.forEach(function (control) {
      var name = control.charAt(0).toUpperCase() + control.slice(1);
```

5 设置地图视图 基于提供的URL或使用默认值，如下所示：

复制

```
map.setView(new ol.View({
  center: [defaultView[0], defaultView[1]],
  zoom: defaultView[2]
}));
```

6 订阅地图视图上的平移和缩放更改，并相应地更新URL：

复制

```
map.getView()
  .on(['change:center', 'change:resolution'], function() {
    window.location.hash = window.location.hash.replace(
      viewHashRegex, 'view=' +
      this.getCenter().join(',') + ',' + this.getZoom()
    );
});
```

怎么运行的...

为了帮助 基于代码的理解，我们假设URL中哈希值的值如下：

复制

```
#/view=-8726204,4937946,12z/controls=[zoom,attribution]
```

URL中的此内容构成地图永久链接的一部分，我们需要基于此书签动态加载地图。

对于视图，该值以 `view=` 开头，之后是x位置，y位置，然后是缩放编号，该缩放编号以逗号分隔并以字母结尾 `z` 。下一个字符，斜杠（ `/` ），充当该字符与下一个值之间的可视中断。

对于控件，该值前面带有， `controls=` 并且控件的名称用方括号（ `[]` ）括起来并以逗号分隔。

现在，让我们看一下JavaScript如何处理此自定义URL：

复制

```
var viewHashRegex = /view=([^z]+)/;
var controlHashRegex = /controls=\[(.+)\]/;
```

我们在变量中存储两个正则表达式以供重用。视图正则表达式将查找 `view=` 字符，然后查找任何字符一次或多次，直到（并排除）该 `z` 字符。 `view=` 字符与字符之间的字符 `z` 以它们自己的组捕获（这就是括号的作用）。

控件正则表达式以类似的方式开始。它匹配 `controls=` 字符，后跟文字 `[` 字符（反斜杠确保文字匹配，因为方括号在正则表达式中具有特殊含义），其次是任意字符一次或多次（被捕获），然后是另一个文字闭合正方形大括号（ `]` ）。

请注意，这些正则表达式尚未经过实战测试；它们仅用于演示目的。有关正则表达式的更多信息，请参阅https://developer.mozilla.org/en/docs/Web/JavaScript/Guide/Regular_Expressions (https://developer.mozilla.org/en/docs/Web/JavaScript/Guide/Regular_Expressions)。

复制

```
var defaultView = [-8726204, 4937946, 12];
```

我们使用默认的中心坐标和缩放级别设置了一个数组。如果未将视图的值指定为URL的一部分，则使用此方法。

我们不会详细介绍地图实例化，但是请注意，该 `controls` 属性已分配了一个空数组（ `controls: []` ）。

让我们在条件检查中剖析核心逻辑的内容：

复制

```
if (window.location.hash) {
  view = window.location.hash.match(viewHashRegex);
  controls = window.location.hash.match(controlHashRegex);
```

我们首先检查一下URL中是否确实存在哈希值。如果不是这种情况，我们可以根据默认值创建地图。

视图和控件的正则表达式的结果分别存储在 `view` 和 `controls` 变量中。

复制

```
if (view) {
  viewArray = view[1].split(',');

  if (viewArray.length === 3) {
    defaultView[0] = viewArray[0];
    defaultView[1] = viewArray[1];
    defaultView[2] = viewArray[2];
  }
}
```

如果是正则表达式使用JavaScript方法 `match` 找不到任何结果，然后返回 `null` 。否则，它将在数组中返回匹配的结果。第一个值是整个匹配项，然后是任何捕获组匹配项。

使用本节开头的示例URL，将返回一个 `match` 包含以下值的数组：

复制

```
["view=-8726204,4937946,12", "-8726204,4937946,12"]
```

考虑到这一点，我们 `split` 在数组的第二项上运行JavaScript 方法，它将 `'-8726204,4937946,12'` 字符串转换为以下数组：

复制

```
["-8726204", "4937946", "12"]
```

我们继续确保该数组的预期长度为三，并 `view` 使用在URL中发现的自定义值覆盖默认数组。

复制

```
if (controls) {
  controlsArray = controls[1].split(',');
  controlsArray.forEach(function (control) {
    var name = control.charAt(0).toUpperCase() + control.slice(1);

    if (typeof ol.control[name] === 'function') {
      map.addControl(new ol.control[name]());
    }
  })
}
```

我们接下来 检查是否 `controls` 返回了正则表达式 `null` 。如果我们有匹配项，我们将再次从中访问返回数组中的第二项 `match` 并将字符串转换为 `controls` 。以我们的URL为例，match返回以下数组：

复制

```
["controls=[zoom,attribution]", "zoom,attribution"]
```

然后，一旦我们 `split` 在第二个数组项中的字符串上运行，我们将产生以下数组：

复制

```
["zoom", "attribution"]
```

然后，我们使用JavaScript `forEach` 方法遍历前面数组中的每个项目。我们期望URL全部使用小写，这意味着 `ol.control.Zoom` OpenLayers控件 `zoom` 在URL中由标识。

当我们尝试调用控件的构造函数时，控件名称必须大写，例如Zoom而不是zoom。为此，我们采用第一个字符（ `charAt(0)` ），在本例中为'z'，将其转换为大写，然后使用JavaScript `slice` 方法将字符串的其余部分连接起来。现在，我们在 `name` 变量中具有值“ Zoom”。

我们通过测试它是否是一种功能来检查该控件是否可用。如果是这样，我们将动态调用控件的构造函数并将其添加到地图中。

复制

```
map.setView(new ol.View({
  center: [defaultView[0], defaultView[1]],
  zoom: defaultView[2]
}));
```

在条件检查之外，我们最终将指定 `view` 给地图，该地图将是默认值或URL中提供的自定义值（将覆盖默认值）。

复制

```
map.getView().on(['change:center', 'change:resolution'], function() {
  window.location.hash = window.location.hash.replace(
    viewHashRegex, 'view=' +
    this.getCenter().join(',') + ',' + this.getZoom()
  );
});
```

最后，我们订阅了地图中心位置（ `change:center` ）以及所有缩放级别调整（ `change:resolution` ）的更改。在处理程序中，我们 `view` 根据当前地图值更新URL中的值（使用相同的正则表达式）。这样可以确保我们的永久链接能够反映最新的可见地图并可以再次共享。

也可以看看

- 第4章 (/book/web_development/9781785287756/4)，**使用事件中**的**侦听矢量层功能的事件**配方 (/book/web_development/9781785287756/4)
- 在**周围的地图视图移动**在配方第1章 (/book/web_development/9781785287756/1)，**Web制图基础知识**

上一节 (/book/web_development/9781785287756/7/ch07lvl1sec68/adding-features-to-the-vec

下一节 (/book/web_development/9781785287756/backindex)