

实施地图图层的进行中指示器

在创造伟大的艺术应用程序，要考虑的最重要的事情之一就是用户体验。一个好的应用程序可以完成它必须要做的事情，但是它可以使用户感到舒适。

使用远程服务器时，会花费大量时间等待数据检索。例如，在使用图块服务时，每次我们更改缩放级别或平移地图时，用户都必须等待一段时间，直到从服务器获取数据并且图块开始渲染。

最好使用微调器图标，进度条或其他熟悉的视觉提示向用户显示一些反馈，以告知用户该应用程序正在运行，但需要一些时间。

此食谱向我们展示了如何通过显示应用程序何时从外部服务加载内容（利用某些层事件）来向用户提供一些反馈。

可以在中找到源代码 `ch04/ch04-map-loading-progress` 。为了方便起见，您还可以在 <https://jsfiddle.net/pjlangley/qd6z4p8b/> (<https://jsfiddle.net/pjlangley/qd6z4p8b/>)查看此食谱。



在 为了创建地图加载 进度栏，执行以下步骤：

- 1 创建具有OpenLayers依赖项的HTML文件。特别是，我们具有以下标记：

复制

```
<div id="js-map"></div>
<div class="progress">
  <div id="js-progress-bar"></div>
</div>
```

- 2 创建一个自定义JavaScript文件并设置栅格源和一些我们将在整个代码中使用的其他全局变量：

复制

```
var rasterSource = new ol.source.MapQuest({layer: 'sat'});
var progressBar = document.getElementById('js-progress-bar');
var tilesLoaded = 0;
var tilesPending = 0;
```

- 3 订阅源上的一些磁贴事件，并在磁贴完成加载后相应地更新进度条：

复制

```
rasterSource.on(['tileloadend', 'tileloaderror'], function() {
  ++tilesLoaded;
  var percentage = Math.round(tilesLoaded / tilesPending * 100);
  progressBar.style.width = percentage + '%';

  if (percentage >= 100) {
    setTimeout(function() {
      progressBar.parentNode.style.opacity = 0;
      progressBar.style.width = 0;
      tilesLoaded = 0;
      tilesPending = 0;
    }, 600);
  }
});
```

- 4 订阅至另一个源事件，并更新进度栏可见性和尚待处理的图块数量：

复制

```
rasterSource.on('tileloadstart', function() {
  progressBar.parentNode.style.opacity = 1;
  ++tilesPending;
});
```

- 5 最后，实例化一个新的地图实例并附加图块源：

复制

```
new ol.Map({
  view: new ol.View({
    zoom: 5,
    center: [-6291073, -1027313]
  }),
  target: 'js-map',
  layers: [
    new ol.layer.Tile({
      source: rasterSource
    })
  ]
});
```

怎么运行的...

我们在地图的右上角放置了一个进度条。我们使用CSS框架Bootstrap来帮助一些样式。有关此内容的CSS不会详细介绍，所以请查看源代码以获取有关其实现方式的更多信息。进度条反映了加载的图块与仍在飞行中的图块的百分比。

我们将重点关注事件订阅以及它们如何影响进度加载栏。这段代码的其余部分应该从以前的食谱中看起来很熟悉：

复制

```
rasterSource.on(['tileloadend', 'tileloaderror'], function() {
  ++tilesLoaded;
  var percentage = Math.round(tilesLoaded / tilesPending * 100);
  progressBar.style.width = percentage + '%';
});
```

使用该 `on` 方法（源继承自 `ol.Observable` ），我们订阅了两种源事件类型（类型为 `ol.source.TileEvent` ）。`tileloadend` 磁贴已被发布时，将发布该事件已成功加载到地图上。`tileloaderror` 当切片加载失败时，将发布该事件。重要的是，我们仍然要捕获失败的事件类型。否则，我们将无法跟踪待处理状态与已加载状态。

我们的处理程序首先增加 `tilesLoaded` 计数（最初设置为 `0` ）。百分比是通过将已加载的瓦片数除以仍待处理的瓦片总数来计算的，然后将其乘以 `100` 以便可以用作进度条宽度的百分比。例如，假设已经加载了两个磁贴，但是八个仍在等待处理。 $2/8 = 0.25$ 。 $0.25 * 100 = 25$ 。

设置进度条的宽度后，我们将百分比字符串连接到数字的末尾，以便可以将其用作有效的CSS值。进度条显示为25%完成。

复制



```
if (percentage >= 100) {
  setTimeout(function() {
    progressBar.parentNode.style.opacity = 0;
    progressBar.style.width = 0;
    tilesLoaded = 0;
    tilesPending = 0;
  }, 600);
}
```

在同一个处理程序中，我们检查加载的砖块百分比是否已达到100%或更高。用户可以在之前的加载会话完成之前通过进一步平移或缩放来触发额外的图块加载。在这种情况下，检查是否大于100%可以解决问题。

在条件块内，我们知道所有图块均已加载。我们重置进度条，使其处于隐藏状态，新的起始宽度为 0，并且磁贴跟踪变量恢复为其原始起始值 0。我们将这种重置包装在 `setTimeout` JavaScript 方法中，因为进度条的宽度会设置为新位置。因此，这使动画有机会在从地图隐藏之前完成100%的操作：

[复制](#)


```
rasterSource.on('tileloadstart', function() {
  progressBar.parentNode.style.opacity = 1;
  ++tilesPending;
});
```

上相反，我们订阅启动磁贴载入时发布的事件。我们的处理程序确保进度条可见（因为至少要加载一个图块），并增加待处理的图块数量。

也可以看看

- [第2章 \(/book/web_development/9781785287756/2\)](/book/web_development/9781785287756/2)，**添加栅格图层中的添加WMS图层配方** (/book/web_development/9781785287756/2)
- [第3章 \(/book/web_development/9781785287756/3\)](/book/web_development/9781785287756/3)，**使用矢量层中的WFS服务器配方中的添加功能** (/book/web_development/9781785287756/3)
- 所述**创建并排侧地图比较配方**
- 所述**矢量层听力设有事件食谱**

◀ [上一节 \(/book/web_development/9781785287756/4/ch04lvl1sec40/creating-a-side-by-side-map\)](/book/web_development/9781785287756/4/ch04lvl1sec40/creating-a-side-by-side-map)

 [节 ▶ \(/book/web_development/9781785287756/4/ch04lvl1sec42/listening-for-the-vector-layer\)](/book/web_development/9781785287756/4/ch04lvl1sec42/listening-for-the-vector-layer)

