链滴

使用 JS 为图片读写扩展信息 - EXIF

作者：xiluotop

原文链接：https://ld246.com/article/1595397846993

来源网站：链滴

使用 JS 为图片写入扩展信息 - EXIF

言归正传，直接切入主题。在某些场景下，需要将一些扩展或隐藏信息写入到图片里，然后下次以直接通过这个图片拿到这些信息做一些事情。如以下一个场景，当从 Cesium 的球上截取了一张图，然后下一次想要将这个图片重新贴回到球的上面，可以使用 Cesium.SingleTileImageryProvider 行单独的一个图层贴片，需要传入图片以及4个东西南北经纬坐标。在通过 Cesium 上导出图片的同我可以获取当前坐标位置然后将该信息存入图片，以后读取图片可以获取存入的内部信息直接进行操。

本次解决该需求的方案采用 JPEG 的 EXIF 因为 JPEG 图片可以存取额外的扩展信息，如 GPS、设、创作人、图像 ID等。

EXIF 介绍

EXIF 又称可交换图像文件格式，是一个专门为数字记录数码照片的属性信息和拍摄数据。简单理就是一种专门用于添加数码信息扩展使用的规范。在 win7 下对 JPEG 格式的图片右键查看属性的详信息可以看到各种 EXIF 的信息，并且可以进行任意的编辑，因此大多只做参考使用。下图为 EXIF 提的部分信息

原文链接：使用 JS 为图片读写扩展信息 - EXIF

项目	信息（举例）
制造厂商	Canon
相机型号	Canon EOS-1Ds Mark III
图象方向	正常（upper-left）
图象分辨率X	300
图象分辨率Y	300
分辨率单位	dpi
Software	AdobePhotoshopCSMacintosh
最后异动时间	2005:10:06 12:53:19
YCbCrPositioning	2
曝光时间	0.00800 (1/125) sec
光圈值	F22
拍摄模式	光圈优先
ISO感光值	100
Exif信息版本	30,32,32,31
图象拍摄时间	2005:09:25 15:00:18
图象存入时间	2005:09:25 15:00:18
曝光补偿（EV+-）	0
测光模式	点测光（Spot）
闪光灯	关闭
镜头实体焦长	12 mm
Flashpix版本	30,31,30,30
图象色域空间	sRGB
图象尺寸X	5616pixel
图象尺寸Y	3744 pixel

piexifjs 使用

piexifjs 是一个用 js 编写的脚本，可以通过图片的 base64、EXIF 代码中获取到想要读取的 EXIF 息，同样也可以向其插入自定义的 EXIF 信息，本次需求主要依靠该 piexifjs 对导出导入的 JPEG 进信息的读写。

- GitHub地址： https://github.com/hMatoba/piexifjs
- 引入方式：浏览器可通过 <script> 标签引入脚本文件，会自动全局注册 piexif 对象，nodejs 需使用 npm i piexifjs 安装开发依赖，并使用 const piexif = require('piexifjs') 引入
- 基本使用：
  - **piexif.load(jpegData)**：通过 JPEG 的base64 或 \xff\xd8 或 Exif 数据获取到 exif 的数据信对象(exifObj)
  - **piexif.dump(exifObj)**：通过 exif 数据对象获取 exif 信息字符串(exifStr)
  - **piexif.insert(exifStr,base64)**：向 base64 中插入图片 EXIF 信息，返回新的 base64 结果

将地理信息存入到 JPEG

假设当前我们已经利用 JS 做好了一张 JPEG 图片并且转换为了 base64，然后目的向 base64 中入地理信息返回新的图片并且导出外部进行图片保存好，为下一步读取提供资源。直接上代码

原文链接：使用 JS 为图片读写扩展信息 - EXIF

```
let base64 = '.....'    // 做好 jpeg 图片的 base64
let gps = {} // 定义一个 gps 数据对象，下面将为 exif 的 GPS数据对象赋值
gps[piexif.GPSIFD.GPSDateStamp] = "GPSPoint=[1&2&3&4]"; // 通过 piexif 为其自定义规范数，其中 'GPSPoint=[1&2&3&4]' 是自定义内容，主要存入 1,2,3,4 后面通过正则获取 GPSPoint 内信息
let exifObj = {    "GPS": gps }; // 定义一个 exifObj 对象
let exifStr = piexif.dump(exifObj); // 通过 piexif 将 exifObj 对象转换为字符串
let resultBase64 = piexif.insert(exifStr, base64); // 将 exif 结果字符串插入到 base64 获取结果
```

结果示例：

```
first: Exif  MM *  
result: Exif  MM *  
[1&2&3&4]
```

使用 JS 读取 JPEG 中的信息

通过上面获取的 resultBase64 读取之前存入的 GPSPoint，代码如下：

```
let nowExifStr = piexif.dump(piexif.load(resultBase64))
let exifStr = nowExifStr.match(/(GPSPoint\=\[=?)(\S*)(?=\])/))    // 通过正则获取目标 exif 信息
if(exifStr) {
    // 如果获取到信息那么该结果正则返回的结果数组中第三个存的应为 1&2&3&4
    exifStr = exifStr[2];
    console.log('get img point', exifStr)
}
```

示例结果：

```
result: Exif  MM *  
get img point 1&2&3&4
```

扩展：获取图片的 base64

通过上面的分析，当前需求都是通过 base64 进行的操作，所以读取问题解决了就要解决另一个题，如何通过 js 获取图片的 base64。由于牵扯地址 URL 等操作，文件协议无法完成，需在 Web 服器下才可使用，以下讨论均以 Web 服务器环境下进行操作。

● 通过 canvas 转换：利用 canvas.getContext("2d").drawImage 使用 img 加载图片后画到 canvas 最后通过 canvas.toDataURL('image/jpeg') 转换为 base64。但此法不可取，因为 toDataURL 相当重新生成一张图，原本的图片格式被重置了，隐藏的地理信息也就消失了

● 通过 input file：input file 可以获取文件 File() 对象，通过 FileReader.readAsDataURL 进行转换该方案多用于用户自定义上传图片获取信息使用，但对本文的需求不够灵活。当前需求是在目录下创一个 image 文件夹，根据路径访问图片然后获取图片的 base64 以及地理坐标然后画到球上面。以两种转换 base64 为网上多数方案，下面是通过查阅相关资料发现的第三种转换方式

Blob 转 Base64

● blob MDN 解释：   Blob对象表示一个不可变、原始数据的类文件对象。Blob 表示的不一定是Jav Script原生格式的数据。File 接口基于Blob，继承了 blob 的功能并将其扩展使其支持用户系统上的件。

● 使用 fetch 请求图片资源：fetch 传入路径的请求资源，成功后会返回一个 blob 此时借用 FileRead r 就可以将 blob 再转换为 base64 也不会改变原文件的内容。代码如下

```
fetch('./test.jpg')
```

```
    .then((response) => {
        if (!response.ok) {
            throw new Error('Network response was not ok');
        }
        return response.blob();
    })
    .then((myBlob) => {
        var reader = new FileReader();
        reader.readAsDataURL(myBlob);
        reader.onload = function (e) {
            var tempBase64 = e.target.result
            var nowExifStr = piexif.dump(piexif.load(tempBase64))
            console.log(nowExifStr)
            var geopoint = (temp = nowExifStr.match(/(GPSPoint\=\[=?)(\S*)(?=\])/)).length == 3 ?
temp[2] :undefined;
            console.log('get fetch image point:', geopoint)
            document.getElementById('img').src = tempbase64
        }
    })
    .catch((error) => {
        console.error('There has been a problem with your fetch operation:', error);
    })
```

输出结果：

```
Exif  MM *  ◘ ┌^% ┘   ┌   →     ┌ " ┐   ‼   (GPSPoint=[1&2&3&4]
get fetch image point: 1&2&3&4
```

总结

□□□□Exif 图片的扩展信息只能用于 JPEG 图片，局限性还是比较大的，而且 Exif 信息可以随意更改，信息安全无法得到保障，因此该功能并不适用有安全验证的场景。仅能做一些自定的规则功能。当然取 JPEG 的常规信息用的也比较多，从手机中拍摄的照片就可以进行读取各种地理、时间、作者等。过要注意图片与 canvas 进行转换过程中，图片已经被更新掉了。所以为保证信息不丢失在图片更新将这些信息保存好，转换完新的 base64 再将先前保存的 exif 重新插入到 base64 再导出。