在 Node.js 中利用 js-xlsx 处理 Excel 文件

2016.01.30 Scarletsky javascript 热度 7440℃

简介

本文介绍用 Node.js 中的 `js-xlsx` 库来处理 Excel 文件。

`js-xlsx` 库是目前 Github 上 star 数量最多的处理 Excel 的库，功能强大，但上手难度稍大。文档有些乱，不适合快速上手。

本文对 `js-xlsx` 库进行一定的总结，并提供几个实用的例子供读者测试，学习，交流。

安装

```
$ npm install xlsx
```

一些概念

在使用这个库之前，先介绍库中的一些概念。

`workbook` 对象，指的是整份 Excel 文档。我们在使用 `js-xlsx` 读取 Excel 文档之后就会获得 `workbook` 对象。

`worksheet` 对象，指的是 Excel 文档中的表。我们知道一份 Excel 文档中可以包含很多张表，而每张表对应的就是 `worksheet` 对象。

`cell` 对象，指的就是 `worksheet` 中的单元格，一个单元格就是一个 `cell` 对象。

它们的关系如下：

```
// workbook
{
    SheetNames: ['sheet1', 'sheet2'],
    Sheets: {
        // worksheet
        'sheet1': {
            // cell
            'A1': { ... },
            // cell
```

```
            // cell
            'A2': { ... },
            ...
        }
    }
}
```

用法

基本用法

1. 用 `XLSX.readFile` 打开 Excel 文件，返回 `workbook`
2. 用 `workbook.SheetNames` 获取表名
3. 用 `workbook.Sheets[xxx]` 通过表名获取表格
4. 按自己的需求去处理表格
5. 生成新的 Excel 文件

具体用法

读取 Excel 文件

```
import XLSX from 'xlsx';
const workbook = XLSX.readFile('someExcel.xlsx', opts);
```

获取 Excel 文件中的表

```
// 获取 Excel 中所有表名
const sheetNames = workbook.SheetNames; // 返回 ['sheet1'
// 根据表名获取对应某张表
const worksheet = workbook.Sheets[sheetNames[0]];
```

通过 `worksheet[address]` 来操作表格，以 `!` 开头的 key 是特殊的字段。

```
// 获取 A1 单元格对象
let a1 = worksheet['A1']; // 返回 { v: 'hello', t: 's',
// 获取 A1 中的值
a1.v // 返回 'hello'
// 获取表的有效范围
worksheet['!ref'] // 返回 'A1:B20'
worksheet['!range'] // 返回 range 对象，{ s: { r: 0, c: 0
// 获取合并过的单元格
```

> **Tips** 事实上，你可以直接通过 `XLSX.utils.sheet_to_json(worksheet)` 获得同样的结果

注意 本例子中假设表的第一行为字段名

```
const headers = {};
const data = [];
const keys = Object.keys(worksheet);
keys
    // 过滤以 ! 开头的 key
    .filter(k => k[0] !== '!')
    // 遍历所有单元格
    .forEach(k => {
        // 如 A11 中的 A
        let col = k.substring(0, 1);
        // 如 A11 中的 11
        let row = parseInt(k.substring(1));
        // 当前单元格的值
        let value = worksheet[k].v;
        // 保存字段名
        if (row === 1) {
            headers[col] = value;
            return;
        }
        // 解析成 JSON
        if (!data[row]) {
            data[row] = {};
        }
        data[row][headers[col]] = value;
    });
console.log(data); // [ { '姓名': 'test1', '年龄': 20 },
```

合并表格

步骤：

1. 读取多份表格
2. 合并数组

Tips: 其实合并表格跟 `XLSX` 没什么关系，只是处理几个数组而已。

3	test3	18

sheet2

id	country	remark
1	China	hello
2	America	world
3	Unkonw	???

```
let sheet1 = XLSX.utils.sheet_to_json(sheet1);
let sheet2 = XLSX.utils.sheet_to_json(sheet2);
// 先合并 sheet1 和 sheet2，再对统一处理
const result = sheet1.concat(sheet2).reduce((prev, next)
    let index = prev.findIndex((elem, i) => elem.id ===
    if (index === -1) {
        return prev.concat(next);
    } else {
        prev[index] = Object.assign({}, prev[index], nex
        return prev;
    }
}, []);
console.log(result);
// [ { id: '1',
//    name: 'test1',
//    age: '30',
//    country: 'China',
//    remark: 'hello' },
//  { id: '2',
//    name: 'test2',
//    age: '20',
//    country: 'America',
//    remark: 'world' },
//  { id: '3',
//    name: 'test3',
//    age: '18',
//    country: 'Unkonw',
//    remark: '???' } ]
```


导出表格

```
`
    SheetNames: ['mySheet'],
    Sheets: {
        'mySheet': {
            '!ref': 'A1:E4', // 必须要有这个范围才能输出，否
            A1: { v: 'id' },
            ...
        }
    }
}
```

```
var _headers = ['id', 'name', 'age', 'country', 'remark'
var _data = [ { id: '1',
                name: 'test1',
                age: '30',
                country: 'China',
                remark: 'hello' },
              { id: '2',
                name: 'test2',
                age: '20',
                country: 'America',
                remark: 'world' },
              { id: '3',
                name: 'test3',
                age: '18',
                country: 'Unkonw',
                remark: '???' } ];
var headers = _headers
                // 为 _headers 添加对应的单元格位置
                // [ { v: 'id', position: 'A1' },
                //   { v: 'name', position: 'B1' },
                //   { v: 'age', position: 'C1' },
                //   { v: 'country', position: 'D1' },
                //   { v: 'remark', position: 'E1' } ]
                .map((v, i) => Object.assign({}, {v: v,
                // 转换成 worksheet 需要的结构
                // { A1: { v: 'id' },
                //   B1: { v: 'name' },
                //   C1: { v: 'age' },
                //   D1: { v: 'country' },
                //   E1: { v: 'remark' } }
                .reduce((prev, next) => Object.assign({}
var data = _data
              // 匹配 headers 的位置，生成对应的单元格数据
              // [ [ { v: '1', position: 'A2' },
              //     { v: 'test1', position: 'B2' },
              //     { v: '30', position: 'C2' },
```

```
                //     { v: 'test3', position: 'B4' },
                //     { v: '18', position: 'C4' },
                //     { v: 'Unkonw', position: 'D4' },
                //     { v: '???', position: 'E4' } ] ]
                .map((v, i) => _headers.map((k, j) => Obje
                // 对刚才的结果进行降维处理（二维数组变成一维数组）
                // [ { v: '1', position: 'A2' },
                //   { v: 'test1', position: 'B2' },
                //   { v: '30', position: 'C2' },
                //   { v: 'China', position: 'D2' },
                //   { v: 'hello', position: 'E2' },
                //   { v: '2', position: 'A3' },
                //   { v: 'test2', position: 'B3' },
                //   { v: '20', position: 'C3' },
                //   { v: 'America', position: 'D3' },
                //   { v: 'world', position: 'E3' },
                //   { v: '3', position: 'A4' },
                //   { v: 'test3', position: 'B4' },
                //   { v: '18', position: 'C4' },
                //   { v: 'Unkonw', position: 'D4' },
                //   { v: '???', position: 'E4' } ]
                .reduce((prev, next) => prev.concat(next))
                // 转换成 worksheet 需要的结构
                //   { A2: { v: '1' },
                //     B2: { v: 'test1' },
                //     C2: { v: '30' },
                //     D2: { v: 'China' },
                //     E2: { v: 'hello' },
                //     A3: { v: '2' },
                //     B3: { v: 'test2' },
                //     C3: { v: '20' },
                //     D3: { v: 'America' },
                //     E3: { v: 'world' },
                //     A4: { v: '3' },
                //     B4: { v: 'test3' },
                //     C4: { v: '18' },
                //     D4: { v: 'Unkonw' },
                //     E4: { v: '???' } }
                .reduce((prev, next) => Object.assign({},
// 合并 headers 和 data
var output = Object.assign({}, headers, data);
// 获取所有单元格的位置
var outputPos = Object.keys(output);
// 计算出范围
var ref = outputPos[0] + ':' + outputPos[outputPos.lengt
// 构建 workbook 对象
var wb = {
    SheetNames: ['mySheet'],
```

参考资料

https://github.com/SheetJS/js-xlsx

http://stackoverflow.com/questions/30859901/parse-xlsx-with-node-and-create-json

支持一下