在客户端导入和导出 Excel 文件

你可以通过客户端的导入和导出方法将 Excel 转为 JSON 或者将 JSON 对象转换为 Excel。

导出的 Excel 文件由于是从浏览器上下载的，将会被锁定。Excel 导出是通过客户端导出的，不用和服务器交互。当你首次打开 Excel 文件的时候，会显示一个警告信息。

> 如果想要查看控件的属性，<!DOCTYPE html> 是必须的。

你可以下载 Excel 文件到本地，或者直接获取 Excel 文件的内容（blob）并将其发送到服务器中。

以下方法在导入和导出时将可用：

- open
- save

下表列出了可以导入或者导出的属性：

种类	描述	导入（Excel 到 JSON）	导出（JSON 到 Excel）
workbook (spread)	标签栏: tabStripVisible, startSheetIndex, tabStripRatio, tabColor	√	√
	滚动条: showHorizontalScrollbar, showVerticalScrollbar	√	√
	工作表: 工作表是否可见，工作表名称	√	√
	引用类型: R1C1 or A1	√	√
	自定义名称	√	√
sharedStrings	用于 sheet 数据的字符串（普通字符串，带空格的字符串）	√	√
theme	颜色方案（Color Scheme）	√	√
	字体方案 (Spread.Sheets 没有具体的字体)	√	√
	格式化方案 (Spread.Sheets 没有格式化方案)	X	√
style	cellStyles: 所有 Spread.Sheets 支持的 cellStyle	√	√
	不同的格式 (在 表格，条件格式和筛选器中使用的格式)	√	√
	tableStyles	√	√
worksheet	rowRangeGroup, colRangeGroup	√	√
	rowCount and columnCount	√	√
	gridline visible, gridline color	√	√
	row header and column header visible	√	√
	缩放	√	√
	选择	√	√
	activeRow, activeColumn	√	√
	冻结 (frozenRowCount, frozenColumnCount)	√	√
	默认 rowHeight, 默认 columnWidth	√	√
	列信息: 列宽，列是否可见，列样式	√	√
	合并的单元格（merge）	√	√
	被保护的工作表	√	√
	行信息: 行高, 行是否可见, 行样式	√	√
	单元格信息: 单元格的值，单元格内的公式，单元格格式	√	√
	自定义名称	√	√
	条件格式	√	√
	批注	√	√
	图片	√	√
	切片器	√	√

	迷你图	√	√
	表格	√	√
	筛选器	√	√
	数据验证器	√	√
	大纲	√	√
	打印设置	√	√

示例代码

下列代码打开并保存了一个 Excel 文件。第一部分的代码是所依赖的 JavaScript 文件：

HTML

```
<script
src="https://cdnjs.cloudflare.com/ajax/libs/FileSaver.js/2014-11-29/File
Saver.min.js"></script>
<link href="./css/gc.spread.sheets.excel2013white.x.xx.xxxx.x.css"
rel="stylesheet"/>
<script src="./scripts/gc.spread.sheets.all.x.xx.xxxx.x.min.js"
type="application/javascript"></script>
<!--For client-side excel i/o-->
<script
src="./scripts/interop/gc.spread.excelio.x.xx.xxxxx.x.min.js"></script>
```

HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>SpreadJS V10 Client Side ExcelIO</title>
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/FileSaver.js/2014-11-29/File
Saver.min.js"></script>
    <link href="./css/gc.spread.sheets.excel2013white.10.x.x.css"
rel="stylesheet"/>
    <script src="./scripts/gc.spread.sheets.all.10.x.x.min.js"
type="application/javascript"></script>
    <!--For client-side excel i/o-->
    <script
src="./scripts/interop/gc.spread.excelio.10.x.x.min.js"></script>
</head>
<body>
<div>
    <input type="file" name="files[]" id="fileDemo"
accept=".xlsx,.xls"/>
    <input type="button" id="loadExcel" value="Import"
onclick="ImportFile()"/>
    <input type="button" class="btn btn-default" id="saveExcel"
value="Export" onclick="ExportFile()"/>
<input type="text" id="exportFileName" placeholder="Export file name"
class="form-control" value="export.xlsx"/>
    <div id="ss" style="width:100%;height:500px"></div>
</div>
</div>
```

```
</body>
<script>
    var workbook, excelIO;
window.onload = function () {
workbook = new GC.Spread.Sheets.Workbook(document.getElementById("ss"));
excelIO = new GC.Spread.Excel.IO();
    }
    function ImportFile() {
        var excelFile = document.getElementById("fileDemo").files[0];
        excelIO.open(excelFile, function (json) {
            var workbookObj = json;
           workbook.fromJSON(workbookObj);
        }, function (e) {
            console.log(e);
        });
    }
    function ExportFile() {
        var fileName = document.getElementById("exportFileName").value;
        if (fileName.substr(-5, 5) !== '.xlsx') {
            fileName += '.xlsx';
        }
        var json = JSON.stringify(workbook.toJSON());
        excelIO.save(json, function (blob) {
            saveAs(blob, fileName);
        }, function (e) {
            console.log(e);
        });
```

```
    }
</script>
</html>
```