TableExport · TableExport

tableexport.v5.travismclarke.com/

TableExport

The simple, easy-to-implement library to export HTML tables to `xlsx`, `xls`, `csv`, and `txt` files.

release v5.0.2 build passing downloads 50k

license Apache-2.0

Docs

Getting Started

Install manually using `<script>` tags

To use this library, include the FileSaver.js library, and TableExport library before the closing `<body>` tag of your HTML document:

```
<script src="FileSaver.js"></script>
 ...
<script src="tableexport.js"></script>
```

Install with Bower

```
$ bower install tableexport.js
```

Install with npm

```
$ npm install tableexport
```

CDN

CDNjs

unpkg

Dependencies

Required:

- FileSaver.js

Optional:

- jQuery (1.2.1 or higher)
- Bootstrap (3.1.0 or higher)

Add-Ons:

In order to provide **Office Open XML SpreadsheetML Format (`.xlsx`)** support, you must include the following third-party library in your project before both FileSaver.js and TableExport.

> xlsx.core.js by *SheetJS*

> Including `xlsx.core.js` is **NOT** necessary if installing with `Bower` or `npm`

```
<script src="xlsx.core.js"></script>
<script src="FileSaver.js"></script>
 ...
<script src="tableexport.js"></script>
```

Older Browsers:

To support legacy browsers (**Chrome** < 20, **Firefox** < 13, **Opera** < 12.10, **IE** < 10, **Safari** < 6) include the Blob.js polyfill before the FileSaver.js script.

> Blob.js by *eligrey* (forked by *clarketm*)

> Including `Blob.js` is **NOT** necessary if installing with `Bower` or `npm`

```
<script src="Blob.js"></script>
<script src="FileSaver.js"></script>
 ...
<script src="tableexport.js"></script>
```

Usage

JavaScript

To use this library, simple call the `TableExport` constructor:

```
new TableExport(document.getElementsByTagName("table"));
```

```
TableExport(document.getElementsByTagName("table"));
```

```
$("table").tableExport();
```

Additional properties can be passed-in to customize the look and feel of your tables, buttons, and exported data.

Notice that by default, TableExport will create export buttons for three different filetypes `xls` , `csv` , `txt` . You can choose which buttons to generate by setting the `formats` property to the filetype(s) of your choice.

```
TableExport(document.getElementsByTagName("table"), {
    headers: true,
    footers: true,
    formats: ['xlsx', 'csv', 'txt'],
    filename: 'id',
    bootstrap: false,
    exportButtons: true,
    position: 'bottom',
    ignoreRows: null,
    ignoreCols: null,
    trimWhitespace: true
});
```

> **Note:** to use the `xlsx` filetype, you must include js-xlsx; reference the `Add-Ons` section.

Properties

Methods

TableExport supports additional methods (**getExportData**, **update**, **reset** and **remove**) to control the `TableExport` instance after creation.

```
var table = TableExport(document.getElementById("export-buttons-table"));
```

`getExportData`

```
var exportData = table.getExportData();
```

`getFileSize`

```
var tableId = 'export-buttons-table';
var XLS = table.CONSTANTS.FORMAT.XLS;


var exportDataXLS = table.getExportData()[tableId][XLS];


var bytesXLS = table.getFileSize(exportDataXLS.data, exportDataXLS.fileExtension);
```

`update`

```
table.update({
    filename: "newFile"
});
```

`reset`

```
table.reset();
```

`remove`

```
table.remove();
```

Settings

Below are some of the popular configurable settings to customize the functionality of the library.

`ignoreCSS`

```
TableExport.prototype.ignoreCSS = ".tableexport-ignore";
```

```
TableExport.prototype.ignoreCSS = [".tableexport-ignore", ".other-ignore-class"];
```

```
$.fn.tableExport.ignoreCSS = ".tableexport-ignore" ;
```

```
$.fn.tableExport.ignoreCSS = [".tableexport-ignore", ".other-ignore-class"] ;
```

`emptyCSS`

```
TableExport.prototype.emptyCSS = ".tableexport-empty";
```

```
TableExport.prototype.emptyCSS = [".tableexport-empty", ".other-empty-class"];
```

```
$.fn.tableExport.emptyCSS = ".tableexport-empty" ;
```

```
$.fn.tableExport.emptyCSS = [".tableexport-empty", ".other-empty-class"];
```

```
TableExport.prototype.charset = "charset=utf-8";


TableExport.prototype.defaultFilename = "myDownload";


TableExport.prototype.defaultButton = "button-default";


TableExport.prototype.bootstrapConfig = ["btn", "btn-default", "btn-toolbar"];


TableExport.prototype.rowDel = "\r\n";


formatConfig: {

    xlsx: {
        defaultClass: 'xlsx',
        buttonContent: 'Export to xlsx',
        mimeType: 'application/vnd.openxmlformats-
officedocument.spreadsheetml.sheet',
        fileExtension: '.xlsx'
    },
    xlsm: {
        defaultClass: 'xlsm',
        buttonContent: 'Export to xlsm',
        mimeType: 'application/vnd.ms-excel.sheet.macroEnabled.main+xml',
        fileExtension: '.xlsm'
    },
    xlsb: {
        defaultClass: 'xlsb',
        buttonContent: 'Export to xlsb',
        mimeType: 'application/vnd.ms-excel.sheet.binary.macroEnabled.main',
        fileExtension: '.xlsb'
    },

    xls: {
        defaultClass: 'xls',
        buttonContent: 'Export to xls',
        separator: '\t',
        mimeType: 'application/vnd.ms-excel',
        fileExtension: '.xls',
        enforceStrictRFC4180: false
    },

    csv: {
        defaultClass: 'csv',
        buttonContent: 'Export to csv',
        separator: ',',
        mimeType: 'text/csv',
        fileExtension: '.csv',
        enforceStrictRFC4180: true
    },

    txt: {
        defaultClass: 'txt',
        buttonContent: 'Export to txt',
        separator: '  ',
        mimeType: 'text/plain',
        fileExtension: '.txt',
        enforceStrictRFC4180: true
    }
},
```

```
TableExport.prototype.formatConfig.xlsx.mimeType = "application/csv"
```

CSS

TableExport packages with customized Bootstrap CSS stylesheets to deliver enhanced table and button styling. These styles can be *enabled* by initializing with the `bootstrap` property set to `true`.

```
TableExport(document.getElementsByTagName("table"), {
    bootstrap: true
});
```

When used alongside Bootstrap, there are four custom classes **`.xlsx`**, **`.xls`**, **`.csv`**, **`.txt`** providing button styling for each of the exportable filetypes.

Browser Support

	Chrome	Firefox	IE	Opera	Safari
Android	✓	✓	-	✓	-
iOS	✓	-	-	-	✓
Mac OSX	✓	✓	-	✓	✓
Windows	✓	✓	✓	✓	✓

> A full list of browser support can be found in the FileSaver.js README. Some legacy browsers may require an additional third-party dependency: Blob.js

Examples

Customizing Properties

Customizing Settings

- `ignoreCSS`
- `emptyCSS`

Miscellaneous

Skeletons

License

TableExport is licensed under the terms of the Apache-2.0 License

Going Forward

TODOs

- [x] Update JSDocs and TypScript definition file.
- [x] Fix bug with **CSV** and **TXT** `ignoreRows` and `ignoreCols` (rows/cols rendered as empty strings rather than being *removed*).
- [x] Reimplement and test the `update` , `reset` , and `remove` **TableExport** prototype properties without requiring jQuery.
- [x] Make jQuery as *peer dependency* and ensure proper **TableExport** rendering in browser, AMD, and CommonJS environments.
- [x] Force jQuery to be an optionally loaded module.
- [x] Use the enhanced SheetJS `xls` , `csv` , and `txt` formats (exposed via `enforceStrictRFC4180` prototype property).
- [x] Allow `ignoreCSS` and `emptyCSS` to work with any `selector|selector[]` instead of solely a single CSS class.
- [x] Ensure (via testing) full consistency and backwards-compatibility for jQuery.
- [ ] Add **Export as PDF** support.

Credits

Special thanks the the following contributors:

- SheetJS - js-xlsx
- Eli Grey - FileSaver.js & Blob.js