



 [simonbengtsson](#) / [jsPDF-AutoTable](#)jsPDF plugin for generating PDF tables with javascript <https://simonbengtsson.github.io/jsPD...>

#jspdf #pdftables #jspdf-plugin #javascript

 320 commits 5 branches 63 releases 13 contributors MIT

Branch: master ▾

















New pull request

Create new file

Upload files

Find file

Clone or download ▾

 simonbengtsson committed on 22 Dec 2017 Merge pull request #310 from tarekis/fix/print-less-than-zero-remaini... 	Latest commit d267119 on 22 Dec 2017
 dist	eliminate scenarios where less than zero remaining lines are printable 2 months ago
 examples	2.3.2 9 months ago
 src	eliminate scenarios where less than zero remaining lines are printable 2 months ago
 test	Update build a year ago
 .gitignore	Update build a year ago
 ISSUE_TEMPLATE.md	Typescript conversion a year ago
 LICENSE.txt	Corrected repository urls 2 years ago
 README.md	Updated README.md 8 months ago
 STORY.md	Update story 2 years ago
 bower.json	Updated to jspdf v1.2.61 2 years ago
 package.json	2.3.2 9 months ago
 samples.png	Fixed some docs for v2 3 years ago
 tsconfig.json	Typescript conversion a year ago
 webpack.config.js	Cell padding fix a year ago

 [README.md](#) Buy Me a Coffee

AutoTable - Table plugin for jsPDF

Generate PDF tables with javascript

Check out the [demo](#) to get an overview of what can be done with this plugin. Example uses include participant tables, start lists, result lists etc.

Theme "striped"

ID	Name	Email	City	Country	Expenses
1	Mr. Verla Schneider	Gaylord_Ortiz56@gmail.com	North Berenice	Hungary	\$308.64
2	Antonette Ferry	Joel.McClure@hotmail.com	East Joeypport	Tajikistan	\$688.22
3	Mrs. Wilber Tillman	Bernard66@gmail.com	Lake Anita	Angola	\$640.36
4	Jazmin Hyatt	Logan_Bednar@gmail.com	Tamiafurt	Lebanon	\$705.99

Theme "grid"

ID	Name	Email	City	Country	Expenses
1	Justus Cassin	Josianne.Moore@hotmail.com	South Marlene	Timor-Leste	\$241.73
2	Luciano Okuneva	Doyle.Boehm19@hotmail.com	Carmeloview	Palau	\$306.26
3	Ernie Feil III	Carmella.McLaughlin@gmail.com	North Mariaborough	Australia	\$919.08
4	Estrella Grady	London_Reynolds80@gmail.com	Kulasberg	Vanuatu	\$869.17

Theme "plain"

ID	Name	Email	City	Country	Expenses
1	Ladarius Brakus	Rozella45@gmail.com	Kozeyhaven	Nauru	\$133.44
2	Josephine O'Conner	Maggie_Crona@yahoo.com	Kilbackhaven	Kazakhstan	\$853.13
3	Ms. Samir Buckridge	Kelvin_Fisher28@hotmail.com	Lueilwitzmouth	Taiwan	\$45.52
4	Miss Travon Gibson	Helena58@hotmail.com	Lake Hortensechester	China	\$905.64

Install

Download and include [jspdf.plugin.autotable.js](#) and [jspdf.min.js](#).

```
<script src="bower_components/jspdf/dist/jspdf.min.js"></script>
<script src="bower_components/jspdf-autotable/dist/jspdf.plugin.autotable.js"></script>
```

You can also get the plugin with a package manager:

- `bower install jspdf-autotable`
- `npm install jspdf-autotable` (only client side usage)

It is also available on cdnjs:

```
<script src="https://cdnjs.cloudflare.com/ajax/libs/jspdf-autotable/2.3.2/jspdf.plugin.autotable.js"></script>
```

Note! If you are using meteor, use the npm release. Do not use the `jspdf:autotable` package on atmosphere as it is currently outdated.

Browser support

Tested with IE10, IE11 and modern browsers (chrome, edge, firefox and safari).

Usage

```
var columns = ["ID", "Name", "Country"];
var rows = [
  [1, "Shaw", "Tanzania", ...],
  [2, "Nelson", "Kazakhstan", ...],
  [3, "Garcia", "Madagascar", ...],
  ...
];
```

```
// Only pt supported (not mm or in)
var doc = new jsPDF('p', 'pt');
doc.autoTable(columns, rows);
doc.save('table.pdf');
```

You can also use it with webpack, requirejs and other module bundlers ([examples](#)).

Usage with options

```
var columns = [
  {title: "ID", dataKey: "id"},
  {title: "Name", dataKey: "name"},
  {title: "Country", dataKey: "country"},
  ...
];
var rows = [
  {"id": 1, "name": "Shaw", "country": "Tanzania", ...},
  {"id": 2, "name": "Nelson", "country": "Kazakhstan", ...},
  {"id": 3, "name": "Garcia", "country": "Madagascar", ...},
  ...
];

// Only pt supported (not mm or in)
var doc = new jsPDF('p', 'pt');
doc.autoTable(columns, rows, {
  styles: {fillColor: [100, 255, 255]},
  columnStyles: {
    id: {fillColor: 255}
  },
  margin: {top: 60},
  addPageContent: function(data) {
    doc.text("Header", 40, 30);
  }
});
doc.save('table.pdf');
```

See other examples in `/examples/examples.js` which is also the source code for the [demo](#) documents.

Usage with Angular 2 (angular cli v1.0.0-beta.14)

- In an angular cli project run `npm install jspdf-autotable --save`
- Add the `jspdf` and `jspdf-autotable` files to the scripts section in `angular-cli.json` (see below)
- Declare jsPDF as a global variable `declare var jsPDF: any;` and use as normal in any component

```
// angular-cli.json
"scripts": [
  "../node_modules/jspdf/dist/jspdf.min.js",
  "../node_modules/jspdf-autotable/dist/jspdf.plugin.autotable.js"
],
```

```
// app.component.ts or any other component
import { Component } from '@angular/core';
```

```
declare var jsPDF: any; // Important
```

```
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
```

```

    title = 'app works!';

    constructor() {
      var doc = new jsPDF('p', 'pt');
      doc.autoTable(columns, data);
      doc.save("table.pdf");
    }
  }
}

```

Options

All options below are used in `examples.js` so be sure to check it out if in doubt.

```

{
  // Styling
  theme: 'striped', // 'striped', 'grid' or 'plain'
  styles: {},
  headerStyles: {},
  bodyStyles: {},
  alternateRowStyles: {},
  columnStyles: {},

  // Properties
  startY: false, // false (indicates margin top value) or a number
  margin: 40, // a number, array or object
  pageBreak: 'auto', // 'auto', 'avoid' or 'always'
  tableWidth: 'auto', // 'auto', 'wrap' or a number,
  showHeader: 'everyPage' // 'everyPage', 'firstPage', 'never',
  tableLineColor: 200, // number, array (see color section below)
  tableLineWidth: 0,

  // Hooks
  createdHeaderCell: function (cell, data) {},
  createdCell: function (cell, data) {},
  drawHeaderRow: function (row, data) {},
  drawRow: function (row, data) {},
  drawHeaderCell: function (cell, data) {},
  drawCell: function (cell, data) {},
  addPageContent: function (data) {}
};

```

Styles

Styles work similar to css and can be overridden by more specific styles. The overriding order is as follows: Default styles <- theme styles <- styles <- headerStyles and bodyStyles <- alternateRowStyles and columnStyles. It is also possible to override specific cell or row styles using for example the `createdCell` hook. Checkout the `Custom style` example for more information.

```

{
  cellPadding: 5, // a number, array or object (see margin below)
  fontSize: 10,
  font: "helvetica", // helvetica, times, courier
  lineColor: 200,
  lineWidth: 0,
  fontStyle: 'normal', // normal, bold, italic, bolditalic
  overflow: 'ellipsis', // visible, hidden, ellipsis or linebreak
  fillColor: false, // false for transparent or a color as described below
  textColor: 20,
  halign: 'left', // left, center, right
  valign: 'middle', // top, middle, bottom
  columnWidth: 'auto' // 'auto', 'wrap' or a number
}

```

All colors can either be specified as a number (255 white and 0 for black) or an array [red, green, blue] e.g. [255, 255, 255].

Every style above can be changed on a cell by cell basis except for `columnWidth`.

Many of the styles has a matching jsPDF set method. For example `linewidth` corresponds to `doc.setLineWidth()`. More information about those can be found in the jsPDF documentation.

Properties

- `startY` Indicates where the table should start to be drawn on the first page (overriding the margin top value). It can be used for example to draw content before the table. Many examples use this option, but the above use case is presented in the `with content` example.
- `margin` Similar to margin in CSS it sets how much spacing it should be around the table on each page. The `startY` option can be used if the margin top value should be different on the first page. The margin option accepts both a number, an array [top, right, bottom, left] and an object {top: 40, right: 40, bottom: 40, left: 40}. If you want to use the default value and only change one side you can specify it like this: {top: 60}.
- `pageBreak` This option defines the behavior of the table when it will span more than one page. If set to 'always' each table will always start on a new page. If set to 'avoid' it will start on a new page if there is not enough room to fit the entire table on the current page. If set to 'auto' it will add a new page only when the next row doesn't fit.
- `tableWidth` This option defines the fixed width of the table if set to a number. If set to 'auto' it will be 100% of width of the page and if set to 'wrap' it will only be as wide as its content is.
- `showHeader` Set to `firstPage`, `everyPage` OR `never`

Hooks

There are 9 different hooks that gets called at various times during the drawing of the table. If applicable, information about the current cell, row or column are provided to the hook function. In addition to that the following general information is always provided in the `data` parameter:

- `pageCount` - The number of pages it currently spans
- `settings` - The user options merged with the default options
- `table` - Information about the table such as the rows, columns and dimensions
- `doc` - The current jsPDF instance
- `cursor` - The position at which the next table cell will be drawn. This can be assigned new values to create column and row spans. Checkout the Colspan and Rowspan example for more information.

OBS! Only the `drawCell` hook can be used with the native style jsPDF style changes such as `doc.setLineWidth`. If you use the other hooks for changing styles, they will be overridden.

Helper functions

- `autoTableHtmlToJson(tableElem, includeHiddenElements)` Use it to generate the javascript objects required for this library from an HTML table (see `from html` example). If `includeHiddenElements` is set to true hidden rows and columns will be included otherwise excluded.
- `doc.autoTableSetDefaults({ ... })` Use for setting default options for all tables on the specific document. Settings and styles will be overridden in the following order `global < document < table`. Hooks will be added and not overridden.
- `jsPDF.autoTableSetDefaults({ ... })` Use for setting global defaults which will be applied for all document and tables.

If you want to know something about the last table that was drawn you can use `doc.autoTable.previous`. It has a `doc.autoTable.previous.finalY` property among other things that has the value of the last printed y coordinate on a page. This can be used to draw text, multiple tables or other content after a table.

Other pdf libraries

- [pdfmake \(javascript\)](#) I much prefer the coding style of jsPDF over pdfmake, however the tables features of pdfmake are great. And pdfmake have proper support for utf-8 which jsPDF lacks.
- [Included jsPDF table plugin](#) No up to date documentation of how to use it (?) and has bugs.
- [fpdf \(php\)](#) and [pdfbox \(java\)](#) No included table features and have to be used server side.

Contributions

Contributions are always welcome, especially on open issues. If you have something major you want to add or change, please post an issue about it first to discuss it further. The workflow for contributing would be something like this:

- Make code changes
- Start watcher with `npm start` (will build files on file changes)
- Test that the examples works in `examples/index.html`
- Commit and submit pull request

Release workflow (write access to repo required)

- Test and commit code changes
- Run `npm version <semver|major|minor|patch> -m <optional-commit-message>`
- Manually verify files and look over the examples
- Deploy with `npm run deploy`