

[Home](#)

CLASSES

[GState](#)[jsPDF](#)

- addFont
- addGState
- addPage
- beginFormObject
- circle
- clip
- deletePage
- doFormObject
- ellipse
- endFormObject
- getCharSpace
- getCreationDate
- getDrawColor
- getFileId
- getFillColor
- getFont
- getFontList
- getFontSize
- getFormObject
- getLineHeightFactor
- getR2L
- getTextColor
- insertPage
- line
- lines
- ltext
- movePage
- output
- path
- rect
- restoreGraphicsState
- roundedRect
- save
- saveGraphicsState
- setCharSpace
- setCreationDate
- setCurrentTransformationMatrix
- setDisplayMode
- setDocumentProperties
- setDrawColor
- setFileId
- setFillColor
- setFont
- setFontSize
- setFontStyle
- setGState
- setLineCap
- setLineDashPattern
- setLineHeightFactor
- setLineJoin
- setLineMiterLimit
- setLineWidth
- setPage
- setR2L
- setTextColor
- text
- triangle

[Matrix](#)

- applyToPoint
- applyToRectangle
- clone

jsPDF



A library to generate PDFs in client-side JavaScript.

You can [catch me on twitter](#): [@MrRio](#) or head over to [my company's website](#) for consultancy.

[Live Demo](#) | [Documentation](#)

Creating your first document

The easiest way to get started is to drop the CDN hosted library into your page:

```
<script src="https://cdnjs.cloudflare.com/ajax/libs/jspdf/1.5.3/jspdf.de
```

Integrity-hash generated by <https://www.srihash.org/>

or can always get latest version via [unpkg](#)

```
<script src="https://unpkg.com/jspdf@latest/dist/jspdf.min.js"></script>
```

Using yarn:

```
yarn add jspdf
```

Using npm:

```
npm install jspdf --save
```

Then you're ready to start making your document:

```
// Default export is a4 paper, portrait, using millimeters for units
var doc = new jsPDF()

doc.text('Hello world!', 10, 10)
doc.save('a4.pdf')
```

If you want to change the paper size, orientation, or units, you can do:

```
// Landscape export, 2x4 inches
var doc = new jsPDF({
  orientation: 'landscape',
  unit: 'in',
  format: [4, 2]
```

[Home](#)**CLASSES**[GState](#)[jsPDF](#)

[addFont](#)
[addGState](#)
[addPage](#)
[beginFormObject](#)
[circle](#)
[clip](#)
[deletePage](#)
[doFormObject](#)
[ellipse](#)
[endFormObject](#)
[getCharSpace](#)
[getCreationDate](#)
[getDrawColor](#)
[getFileId](#)
[getFillColor](#)
[getFont](#)
[getFontList](#)
[getFontSize](#)
[getFormObject](#)
[getLineHeightFactor](#)
[getR2L](#)
[getTextColor](#)
[insertPage](#)
[line](#)
[lines](#)
[ltext](#)
[movePage](#)
[output](#)
[path](#)
[rect](#)
[restoreGraphicsState](#)
[roundedRect](#)
[save](#)
[saveGraphicsState](#)
[setCharSpace](#)
[setCreationDate](#)
[setCurrentTransformationMatrix](#)
[setDisplayMode](#)
[setDocumentProperties](#)
[setDrawColor](#)
[setFileId](#)
[setFillColor](#)
[setFont](#)
[setFontSize](#)
[setFontStyle](#)
[setGState](#)
[setLineCap](#)
[setLineDashPattern](#)
[setLineHeightFactor](#)
[setLineJoin](#)
[setLineMiterLimit](#)
[setLineWidth](#)
[setPage](#)
[setR2L](#)
[setTextColor](#)
[text](#)
[triangle](#)

[Matrix](#)

[applyToPoint](#)
[applyToRectangle](#)
[clone](#)

}})

```
doc.text('Hello world!', 1, 1)
doc.save('two-by-four.pdf')
```

Use of UTF-8 / TTF:

The 14 standard fonts in PDF are limited to the ASCII-codepage. If you want to use UTF-8 you have to integrate a custom font, which provides the needed glyphs. jsPDF supports .ttf-files. So if you want to have for example chinese text in your pdf, your font has to have the necessary chinese glyphs. So check if your font supports the wanted glyphs or else it will show a blank space instead of the text.

To add the font to jsPDF use our fontconverter in [/fontconverter/fontconverter.html](#) . The fontconverter will create a js-file with the content of the provided ttf-file as base64 encoded string and additional code for jsPDF. You just have to add this generated js-File to your project. You are then ready to go to use setFont-method in your code and write your UTF-8 encoded text.

Angular/Webpack/React/etc. Configuration:

If you are using Webpack (including managed cli tools like angular-cli or create-react-app) you can import like this:

```
import * as jsPDF from 'jspdf'
```

In some frameworks you have to import jsPDF like this:

```
import jsPDF from 'jspdf';
```

You can add jsPDF to your meteor-project as follows:

```
meteor add jspdf:core
```

Support

Please check if your question is already handled at Stackoverflow <https://stackoverflow.com/questions/tagged/jspdf>. Feel free to ask a question there with the tag `jspdf` .

Feature requests, bug reports etc. are very welcome as issues. Note that bug reports should follow these guidelines:

- A bug should be reported as an [mcve](#)
- Make sure code is properly indented and [formatted](#) (Use ``` around code blocks)
- Provide a runnable example.
- Try to make sure and show in your issue that the issue is actually related to jspdf and not your framework of choice your setup.

[Home](#)**CLASSES**[GState](#)[jsPDF](#)

[addFont](#)
[addGState](#)
[addPage](#)
[beginFormObject](#)
[circle](#)
[clip](#)
[deletePage](#)
[doFormObject](#)
[ellipse](#)
[endFormObject](#)
[getCharSpace](#)
[getCreationDate](#)
[getDrawColor](#)
[getFileId](#)
[getFillColor](#)
[getFont](#)
[getFontList](#)
[getFontSize](#)
[getFormObject](#)
[getLineHeightFactor](#)
[getR2L](#)
[getTextColor](#)
[insertPage](#)
[line](#)
[lines](#)
[ltext](#)
[movePage](#)
[output](#)
[path](#)
[rect](#)
[restoreGraphicsState](#)
[roundedRect](#)
[save](#)
[saveGraphicsState](#)
[setCharSpace](#)
[setCreationDate](#)
[setCurrentTransformationMatrix](#)
[setDisplayMode](#)
[setDocumentProperties](#)
[setDrawColor](#)
[setFileId](#)
[setFillColor](#)
[setFont](#)
[setFontSize](#)
[setFontStyle](#)
[setGState](#)
[setLineCap](#)
[setLineDashPattern](#)
[setLineHeightFactor](#)
[setLineJoin](#)
[setLineMiterLimit](#)
[setLineWidth](#)
[setPage](#)
[setR2L](#)
[setTextColor](#)
[text](#)
[triangle](#)

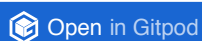
[Matrix](#)

[applyToPoint](#)
[applyToRectangle](#)
[clone](#)

Contributing

Build the library with `npm run build`. This will fetch all dependencies and then compile the `dist` files. To see the examples locally you can start a web server with `npm start` and go to `localhost:8000`.

Alternatively, you can build jsPDF using these commands in a readily configured online workspace:



Credits

- Big thanks to Daniel Dotsenko from [Willow Systems Corporation](#) for making huge contributions to the codebase.
- Thanks to Ajaxian.com for [featuring us back in 2009](#).
- Our special thanks to GH Lee ([sphilee](#)) for programming the ttf-file-support and providing a large and long sought after feature
- Everyone else that's contributed patches or bug reports. You rock.

License (MIT)

Copyright (c) 2010-2017 James Hall, <https://github.com/MrRio/jsPDF>

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Documentation generated by [JSDoc 3.5.5](#) on Tue Apr 02 2019 00:25:07 GMT+0200 (GMT+02:00) using the [docdash](#) theme.