Products Pricing Documentation

npm

Sign Up Sign In

Search packages Search

jspdf TS

2.5.1 • Public • Published 2 months ago

Readme

Explore BETA

8 Dependencies

800 Dependents

24 Versions

jsPDF

Continous Integration passing | maintainability ? | test coverage ? | license MIT

lgtm alerts 91 | code quality: js/ts C | Gitpod ready-to-code

A library to generate PDFs in JavaScript.

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

jsPDF is now co-maintained by yWorks - the diagramming experts.

Live Demo | Documentation

Install

Recommended: get jsPDF from npm:

```
npm install jspdf --save
# or
yarn add jspdf
```

Alternatively, load it from a CDN:

```
<script src="https://cdnjs.cloudflare.com/ajax/libs/jspdf/2.5.1/jsp
```

Or always get latest version via **unpkg**

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

The `dist` folder of this package contains different kinds of files:

- **jspdf.es.*.js**: Modern ES2015 module format.
- **jspdf.node.*.js**: For running in Node. Uses file operations for loading/saving files instead of browser APIs.
- **jspdf.umd.*.js**: UMD module format. For AMD or script-tag loading.
- **polyfills*.js**: Required polyfills for older browsers like Internet Explorer. The es variant simply imports all required polyfills from `core-js`, the umd variant is self-contained.

Usually it is not necessary to specify the exact file in the import statement. Build tools or Node automatically figure out the right file, so importing "jspdf" is enough.

Usage

Then you're ready to start making your document:

```
import { jsPDF } from "jspdf";

// Default export is a4 paper, portrait, using millimeters for unit
const 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, 2×4 inches
const doc = new jsPDF({
  orientation: "landscape",
  unit: "in",
  format: [4, 2]
});

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

Running in Node.js

```
const { jsPDF } = require("jspdf"); // will automatically load the

const doc = new jsPDF();
doc.text("Hello world!", 10, 10);
doc.save("a4.pdf"); // will save the file in the current working di
```

Other Module Formats

▶ AMD

▶ Globals

Optional dependencies

Some functions of jsPDF require optional dependencies. E.g. the `html` method, which depends on `html2canvas` and, when supplied with a string HTML document, `dompurify`. JsPDF loads them dynamically when required (using the respective module format, e.g. dynamic imports). Build tools like Webpack will automatically create separate chunks for each of the optional dependencies. If your application does not use any of the optional dependencies, you can prevent Webpack from generating the chunks by defining them as external dependencies:

```
// webpack.config.js
module.exports = {
  // ...
  externals: {
    // only define the dependencies you are NOT using as externals!
    canvg: "canvg",
    html2canvas: "html2canvas",
    dompurify: "dompurify"
  }
};
```

In **Vue CLI** projects, externals can be defined via the **configureWebpack** or **chainWebpack** properties of the `vue.config.js` file (needs to be created, first, in fresh projects).

In **Angular** projects, externals can be defined using **custom webpack builders**.

In **React** (`create-react-app`) projects, externals can be defined by either using **react-app-rewired** or ejecting.

TypeScript/Angular/Webpack/React/etc. Configuration:

jsPDF can be imported just like any other 3rd party library. This works with all major toolkits and frameworks. jsPDF also offers a typings file for TypeScript projects.

```
import { jsPDF } from "jspdf";
```

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

```
meteor add jspdf:core
```

Polyfills

jsPDF requires modern browser APIs in order to function. To use jsPDF in older browsers like Internet Explorer, polyfills are required. You can load all required polyfills as follows:

```
import "jspdf/dist/polyfills.es.js";
```

Alternatively, you can load the prebundled polyfill file. This is not recommended, since you might end up loading polyfills multiple times. Might still be nifty for small applications or quick POCs.

```
<script src="https://cdnjs.cloudflare.com/ajax/libs/jspdf/2.5.1/poly
```

Use of Unicode Characters / UTF-8:

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 garbled characters instead of the right 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.

Alternatively you can just load the content of the *.ttf file as a binary string using `fetch` or `XMLHttpRequest` and add the font to the PDF file:

```
const doc = new jsPDF();

const myFont = ... // load the *.ttf font file as binary string

// add the font to jsPDF
```

```
doc.addFileToVFS("MyFont.ttf", myFont);
doc.addFont("MyFont.ttf", "MyFont", "normal");
doc.setFont("MyFont");
```

Advanced Functionality

Since the merge with the yWorks fork there are a lot of new features. However, some of them are API breaking, which is why there is an API-switch between two API modes:

- In "compat" API mode, jsPDF has the same API as MrRio's original version, which means full compatibility with plugins. However, some advanced features like transformation matrices and patterns won't work. This is the default mode.
- In "advanced" API mode, jsPDF has the API you're used from the yWorks-fork version. This means the availability of all advanced features like patterns, FormObjects, and transformation matrices.

You can switch between the two modes by calling

```
doc.advancedAPI(doc => {
  // your code
});
// or
doc.compatAPI(doc => {
  // your code
});
```

JsPDF will automatically switch back to the original API mode after the callback has run.

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.

Contributing

jsPDF cannot live without help from the community! If you think a feature is missing or you found a bug, please consider if you can spare one or two hours and prepare a pull request. If you're simply interested in this project and want to help, have a look at the open issues, especially those labeled with "bug".

You can find information about building and testing jsPDF in the **contribution guide**

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)

Keywords

pdf

Install

```
> npm i jspdf
```

Repository

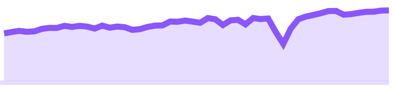
github.com/MrRio/jsPDF

Homepage

github.com/mrrio/jspdf

Weekly Downloads

647,037



Version

2.5.1

License

MIT

Unpacked Size

14.7 MB

Total Files

18

Issues

110

Pull Requests

11

Last publish

2 months ago

Collaborators

>_Try on RunKit

Report malware

Support

Help

Advisories

Status

Contact npm

Company

About

Blog

Press

Terms & Policies

Policies

Terms of Use

Code of Conduct

Privacy