react-pdf TS

9.2.1 • Public • Published 5 months ago

Readme

Code Beta

8 Dependencies

980 Dependents

148 Versions

npm v9.2.1 downloads 76M CI passing

React-PDF

Display PDFs in your React app as easily as if they were images.

Lost?

This package is used to *display* existing PDFs. If you wish to *create* PDFs using React, you may be looking for @react-pdf/renderer.

tl;dr

- Install by executing `npm install react-pdf` or `yarn add react-pdf`.
- Import by adding `import { Document } from 'react-pdf'`.

- Use by adding `<Document file="..." />`. `file` can be a URL, base64 content, Uint8Array, and more.
- Put `<Page />` components inside `<Document />` to render pages.

Demo

A minimal demo page can be found in `sample` directory.

Online demo is also available!

Before you continue

React-PDF is under constant development. This documentation is written for React-PDF 9.x branch. If you want to see documentation for other versions of React-PDF, use dropdown on top of GitHub page to switch to an appropriate tag. Here are quick links to the newest docs from each branch:

- v8.x
- v7.x
- v6.x
- v5.x
- v4.x
- v3.x
- v2.x
- v1.x

Getting started

Compatibility

Browser support

React-PDF supports all modern browsers. It is tested with the latest versions of Chrome, Edge, Safari, Firefox, and Opera.

The following browsers are supported out of the box in React-PDF v9:

- Chrome ≥119
- Edge ≥119
- Safari ≥17.4
- Firefox ≥121

You may extend the list of supported browsers by providing additional polyfills (e.g. for `Array.prototype.at` , `Promise.allSettled` or `Promise.withResolvers`) and either configuring your bundler to transpile `pdfjs-dist` or using **legacy PDF.js worker**.

If you need to support older browsers, you will need to use React-PDF v6 or earlier.

React

To use the latest version of React-PDF, your project needs to use React 16.8 or later.

If you use an older version of React, please refer to the table below to a find suitable React-PDF version.

React version	Newest compatible React-PDF version
≥16.8	latest
≥16.3	5.x
≥15.5	4.x

Preact

React-PDF may be used with Preact.

Installation

Add React-PDF to your project by executing `npm install react-pdf` or `yarn add react-pdf` .

Next.js

If you use Next.js without Turbopack enabled, add the following to your `next.config.js` :

```
module.exports = {
+ webpack: (config) => {
+   config.resolve.alias.canvas = false;

+   return config;
+ },
}
```

If you use Next.js with Turbopack enabled, add `empty-module.ts` file:

```
export default {};
```

and add the following to your `next.config.js`:

```
module.exports = {
+ experimental: {
+   turbo: {
+     resolveAlias: {
+       canvas: './empty-module.ts',
+     },
+   },
+ },
};
```

If you use Next.js prior to v15 (v15.0.0-canary.53, specifically), you may need to add the following to your `next.config.js`:

```
module.exports = {
+ swcMinify: false,
}
```

Configure PDF.js worker

For React-PDF to work, PDF.js worker needs to be provided. You have several options.

Import worker (recommended)

For most cases, the following example will work:

```
import { pdfjs } from 'react-pdf';

pdfjs.GlobalWorkerOptions.workerSrc = new URL(
  'pdfjs-dist/build/pdf.worker.min.mjs',
  import.meta.url,
).toString();
```

> [!NOTE] In Next.js:
>
> - Using App Router, make sure to add `'use client';` to the top of the file.
> - Using Pages Router, make sure to disable SSR when importing the component you're using this code in.

> [!NOTE] pnpm requires an `.npmrc` file with `public-hoist-pattern[]=pdfjs-dist` for this to work.

▶ See more examples

Copy worker to public directory

You will have to make sure on your own that `pdf.worker.mjs` file from `pdfjs-dist/build` is copied to your project's output folder.

For example, you could use a custom script like:

```
import path from 'node:path';
import fs from 'node:fs';

const pdfjsDistPath = path.dirname(require.resolve('pdfjs-dist/package
const pdfWorkerPath = path.join(pdfjsDistPath, 'build', 'pdf.worker.mj
fs.cpSync(pdfWorkerPath, './dist/pdf.worker.mjs', { recursive: true })
```

Use external CDN

```
import { pdfjs } from 'react-pdf';

pdfjs.GlobalWorkerOptions.workerSrc = `//unpkg.com/pdfjs-dist@${pdfjs.
```

Legacy PDF.js worker

If you need to support older browsers, you may use legacy PDF.js worker. To do so, follow the instructions above, but replace `/build/` with `legacy/build/` in PDF.js worker import path, for example:

```
 pdfjs.GlobalWorkerOptions.workerSrc = new URL(
-  'pdfjs-dist/build/pdf.worker.min.mjs',
+  'pdfjs-dist/legacy/build/pdf.worker.min.mjs',
   import.meta.url,
 ).toString();
```

or:

```
-pdfjs.GlobalWorkerOptions.workerSrc = `//unpkg.com/pdfjs-dist@${pdfjs
+pdfjs.GlobalWorkerOptions.workerSrc = `//unpkg.com/pdfjs-dist@${pdfjs
```

Usage

Here's an example of basic usage:

```
import { useState } from 'react';
import { Document, Page } from 'react-pdf';
```

```
function MyApp() {
  const [numPages, setNumPages] = useState<number>();
  const [pageNumber, setPageNumber] = useState<number>(1);

  function onDocumentLoadSuccess({ numPages }: { numPages: number }):
    setNumPages(numPages);
  }

  return (
    <div>
      <Document file="somefile.pdf" onLoadSuccess={onDocumentLoadSucce
        <Page pageNumber={pageNumber} />
      </Document>
      <p>
        Page {pageNumber} of {numPages}
      </p>
    </div>
  );
}
```

Check the **sample directory** in this repository for a full working example. For more examples and more advanced use cases, check **Recipes** in **React-PDF Wiki**.

Support for annotations

If you want to use annotations (e.g. links) in PDFs rendered by React-PDF, then you would need to include stylesheet necessary for annotations to be correctly displayed like so:

```
import 'react-pdf/dist/Page/AnnotationLayer.css';
```

Support for text layer

If you want to use text layer in PDFs rendered by React-PDF, then you would need to include stylesheet necessary for text layer to be correctly displayed like so:

```
import 'react-pdf/dist/Page/TextLayer.css';
```

Support for non-latin characters

If you want to ensure that PDFs with non-latin characters will render perfectly, or you have encountered the following warning:

```
Warning: The CMap "baseUrl" parameter must be specified, ensure that the
```

then you would also need to include cMaps in your build and tell React-PDF where they are.

Copying cMaps

First, you need to copy cMaps from `pdfjs-dist` (React-PDF's dependency - it should be in your `node_modules` if you have React-PDF installed). cMaps are located in `pdfjs-dist/cmaps`.

Vite

Add **vite-plugin-static-copy** by executing `npm install vite-plugin-static-copy --save-dev` or `yarn add vite-plugin-static-copy --dev` and add the following to your Vite config:

```
+import path from 'node:path';
+import { createRequire } from 'node:module';

-import { defineConfig } from 'vite';
+import { defineConfig, normalizePath } from 'vite';
+import { viteStaticCopy } from 'vite-plugin-static-copy';

+const require = createRequire(import.meta.url);
+
+const pdfjsDistPath = path.dirname(require.resolve('pdfjs-dist/packag
+const cMapsDir = normalizePath(path.join(pdfjsDistPath, 'cmaps'));

export default defineConfig({
  plugins: [
+   viteStaticCopy({
+     targets: [
+       {
+         src: cMapsDir,
+         dest: '',
+       },
+     ],
+   }),
  ]
});
```

Webpack

Add `copy-webpack-plugin` by executing `npm install copy-webpack-plugin --save-dev` or `yarn add copy-webpack-plugin --dev` and add the following to your Webpack config:

```
+import path from 'node:path';
+import CopyWebpackPlugin from 'copy-webpack-plugin';

+const pdfjsDistPath = path.dirname(require.resolve('pdfjs-dist/packag
+const cMapsDir = path.join(pdfjsDistPath, 'cmaps');

module.exports = {
  plugins: [
+   new CopyWebpackPlugin({
+     patterns: [
+       {
+         from: cMapsDir,
+         to: 'cmaps/'
+       },
+     ],
+   }),
  ],
};
```

Other tools

If you use other bundlers, you will have to make sure on your own that cMaps are copied to your project's output folder.

For example, you could use a custom script like:

```
import path from 'node:path';
import fs from 'node:fs';

const pdfjsDistPath = path.dirname(require.resolve('pdfjs-dist/package
const cMapsDir = path.join(pdfjsDistPath, 'cmaps');

fs.cpSync(cMapsDir, 'dist/cmaps/', { recursive: true });
```

Setting up React-PDF

Now that you have cMaps in your build, pass required options to Document component by using `options` prop, like so:

```
// Outside of React component
const options = {
  cMapUrl: '/cmaps/',
};

// Inside of React component
<Document options={options} />;
```

> [!NOTE] Make sure to define `options` object outside of your React component, and use `useMemo` if you can't.

Alternatively, you could use cMaps from external CDN:

```
// Outside of React component
import { pdfjs } from 'react-pdf';

const options = {
  cMapUrl: `https://unpkg.com/pdfjs-dist@${pdfjs.version}/cmaps/`,
};

// Inside of React component
<Document options={options} />;
```

Support for standard fonts

If you want to support PDFs using standard fonts (deprecated in PDF 1.5, but still around), ot you have encountered the following warning:

```
The standard font "baseUrl" parameter must be specified, ensure that the
```

then you would also need to include standard fonts in your build and tell React-PDF where they are.

Copying fonts

First, you need to copy standard fonts from `pdfjs-dist` (React-PDF's dependency - it should be in your `node_modules` if you have React-PDF installed). Standard fonts are located in `pdfjs-dist/standard_fonts`.

Vite

Add **`vite-plugin-static-copy`** by executing `npm install vite-plugin-static-copy --save-dev` or `yarn add vite-plugin-static-copy --dev` and add the following to your Vite config:

```
+import path from 'node:path';
+import { createRequire } from 'node:module';
```

```diff
-import { defineConfig } from 'vite';
+import { defineConfig, normalizePath } from 'vite';
+import { viteStaticCopy } from 'vite-plugin-static-copy';

+const require = createRequire(import.meta.url);
+const standardFontsDir = normalizePath(
+  path.join(path.dirname(require.resolve('pdfjs-dist/package.json')),
+);

export default defineConfig({
  plugins: [
+   viteStaticCopy({
+     targets: [
+       {
+         src: standardFontsDir,
+         dest: '',
+       },
+     ],
+   }),
  ]
});
```

Webpack

Add `copy-webpack-plugin` by executing `npm install copy-webpack-plugin --save-dev` or `yarn add copy-webpack-plugin --dev` and add the following to your Webpack config:

```diff
+import path from 'node:path';
+import CopyWebpackPlugin from 'copy-webpack-plugin';

+const standardFontsDir = path.join(path.dirname(require.resolve('pdfj
 module.exports = {
   plugins: [
+    new CopyWebpackPlugin({
+      patterns: [
+        {
+          from: standardFontsDir,
+          to: 'standard_fonts/'
+        },
+      ],
+    }),
   ],
 };
```

Other tools

If you use other bundlers, you will have to make sure on your own that standard fonts are copied to your project's output folder.

For example, you could use a custom script like:

```
import path from 'node:path';
import fs from 'node:fs';

const pdfjsDistPath = path.dirname(require.resolve('pdfjs-dist/package
const standardFontsDir = path.join(pdfjsDistPath, 'standard_fonts');

fs.cpSync(standardFontsDir, 'dist/standard_fonts/', { recursive: true
```

Setting up React-PDF

Now that you have standard fonts in your build, pass required options to Document component by using `options` prop, like so:

```
// Outside of React component
const options = {
  standardFontDataUrl: '/standard_fonts/',
};

// Inside of React component
<Document options={options} />;
```

> [!NOTE] Make sure to define `options` object outside of your React component, and use `useMemo` if you can't.

Alternatively, you could use standard fonts from external CDN:

```
// Outside of React component
import { pdfjs } from 'react-pdf';

const options = {
  standardFontDataUrl: `https://unpkg.com/pdfjs-dist@${pdfjs.version}/
};

// Inside of React component
<Document options={options} />;
```

User guide

Document

Loads a document passed using `file` prop.

Props

Prop name	Description	Default value	
className	Class name(s) that will be added to rendered element along with the	n/a	• String: "cust class-

Prop name	Description	Default value	
	default `react-pdf__Document`.		• Array of `["cus class-`
error	What the component should display in case of an error.	`"Failed to load PDF file."`	• String: `"An e` • React el `<p>An` • Functio `this.`
externalLinkRel	Link rel for links rendered in annotations.	`"noopener noreferrer nofollow"`	One of vali • `"noop` • `"nore` • `"nofo` • `"noop`
externalLinkTarget	Link target for external links rendered in annotations.	unset, which means that default behavior will be used	One of vali • `"_sel` • `"_bla` • `"_par` • `"_top`
file	What PDF should be displayed. Its value can be an URL, a file (imported using `import … from …` or from file input form	n/a	• URL: `"http` • File: `impor`

Prop name	Description	Default value	
	element), or an object with parameters (`url` - URL; `data` - data, preferably Uint8Array; `range` - PDFDataRangeTransport. **Warning**: Since equality check (`===`) is used to determine if `file` object has changed, it must be memoized by setting it in component's state, `useMemo` or other similar technique.		`'../st` `sampl` • Paramet `{ url` `'https` `}`
imageResourcesPath	The path used to prefix the src attributes of annotation SVGs.	n/a (pdf.js will fallback to an empty string)	`"/publi`
inputRef	A prop that behaves like ref, but it's passed to main `<div>` rendered by `<Document>` component.	n/a	• Functio `(ref)` `ref; }` • Ref crea `this.` `…` `input` • Ref crea `const` `…` `input`

Prop name	Description	Default value	
loading	What the component should display while loading.	`"Loading PDF…"`	• String: `"Plea` • React el `<p>Pl` • Functio `this.`
noData	What the component should display in case of no data.	`"No PDF file specified."`	• String: `"Plea` • React el `<p>Pl` • Functio `this.`
onItemClick	Function called when an outline item or a thumbnail has been clicked. Usually, you would like to use this callback to move the user wherever they requested to.	n/a	`({ dest => aler page '`
onLoadError	Function called in case of an error while loading a document.	n/a	`(error) loading error.m`
onLoadProgress	Function called, potentially multiple times, as the loading progresses.	n/a	`({ load alert(' (loaded`

Prop name	Description	Default value	
onLoadSuccess	Function called when the document is successfully loaded.	n/a	`(pdf) =` `' + pdf`
onPassword	Function called when a password-protected PDF is loaded.	Function that prompts the user for password.	`(callba` `callbac`
onSourceError	Function called in case of an error while retrieving document source from `file` prop.	n/a	`(error)` `retriev` `error.m`
onSourceSuccess	Function called when document source is successfully retrieved from `file` prop.	n/a	`() => a` `retriev`
options	An object in which additional parameters to be passed to PDF.js can be defined. Most notably: • `cMapUrl`; • `httpHeaders` - custom request headers, e.g. for authorization); • `withCredentials` - a boolean to indicate whether or not to include cookies in the request (defaults to `false`)	n/a	`{ cMapU`

Prop name	Description	Default value	
	For a full list of possible parameters, check PDF.js documentation on DocumentInitParameters. **Note**: Make sure to define options object outside of your React component, and use `useMemo` if you can't.		
renderMode	Rendering mode of the document. Can be `"canvas"`, `"custom"` or `"none"`. If set to `"custom"`, `customRenderer` must also be provided.	`"canvas"`	`"custom`
rotate	Rotation of the document in degrees. If provided, will change rotation globally, even for the pages which were given `rotate` prop of their own. `90` = rotated to the right, `180` = upside down, `270` = rotated to the left.	n/a	`90`

Page

Displays a page. Should be placed inside `<Document />`. Alternatively, it can have `pdf` prop passed, which can be obtained from `<Document />`'s `onLoadSuccess` callback function, however some advanced functions like rendering annotations and linking between pages inside a document may not be working correctly.

Props

Prop name	Description	
canvasBackground	Canvas background color. Any valid `canvas.fillStyle` can be used.	n/a
canvasRef	A prop that behaves like ref, but it's passed to `<canvas>` rendered by `<Canvas>` component.	n/a
className	Class name(s) that will be added to rendered element along with the default `react-pdf__Page`.	n/a
customRenderer	Function that customizes how a page is rendered. You must	n/a

Prop name	Description	D
	set `renderMode` to `"custom"` to use this prop.	
customTextRenderer	Function that customizes how a text layer is rendered.	n/a
devicePixelRatio	The ratio between physical pixels and device-independent pixels (DIPs) on the current device.	`window.`
error	What the component should display in case of an error.	`"Failed page."`
height	Page height. If neither `height` nor `width` are defined, page will be rendered at the size defined in PDF. If you define `width` and `height` at the same time, `height` will be ignored. If you define `height` and `scale` at the same time, the height will be multiplied by a given factor.	Page's defa
imageResourcesPath	The path used to prefix the src attributes of annotation SVGs.	n/a (pdf.js empty strir

Prop name	Description	I
inputRef	A prop that behaves like ref, but it's passed to main `<div>` rendered by `<Page>` component.	n/a
loading	What the component should display while loading.	`"Loadin`
noData	What the component should display in case of no data.	`"No pag`
onGetAnnotationsError	Function called in case of an error while loading annotations.	n/a

Prop name	Description	D
onGetAnnotationsSuccess	Function called when annotations are successfully loaded.	n/a
onGetStructTreeError	Function called in case of an error while loading structure tree.	n/a
onGetStructTreeSuccess	Function called when structure tree is successfully loaded.	n/a
onGetTextError	Function called in case of an error while loading text layer items.	n/a
onGetTextSuccess	Function called when text layer items are successfully loaded.	n/a
onLoadError	Function called in case of an error while loading the page.	n/a
onLoadSuccess	Function called when the page is successfully loaded.	n/a
onRenderAnnotationLayerError	Function called in case of an error while rendering the annotation layer.	n/a
onRenderAnnotationLayerSuccess	Function called when annotations are successfully rendered on the screen.	n/a

Prop name	Description	D
onRenderError	Function called in case of an error while rendering the page.	n/a
onRenderSuccess	Function called when the page is successfully rendered on the screen.	n/a
onRenderTextLayerError	Function called in case of an error while rendering the text layer.	n/a
onRenderTextLayerSuccess	Function called when the text layer is successfully rendered on the screen.	n/a
pageIndex	Which page from PDF file should be displayed, by page index. Ignored if `pageNumber` prop is provided.	`0`
pageNumber	Which page from PDF file should be displayed, by page number. If provided, `pageIndex` prop will be ignored.	`1`
pdf	pdf object obtained from `<Document />`'s `onLoadSuccess` callback function.	(automatic parent <D
renderAnnotationLayer	Whether annotations (e.g. links) should be rendered.	`true`

Prop name	Description	D
renderForms	Whether forms should be rendered. `renderAnnotationLayer` prop must be set to `true`.	`false`
renderMode	Rendering mode of the document. Can be `"canvas"`, `"custom"` or `"none"`. If set to `"custom"`, `customRenderer` must also be provided.	`"canvas`
renderTextLayer	Whether a text layer should be rendered.	`true`
rotate	Rotation of the page in degrees. `90` = rotated to the right, `180` = upside down, `270` = rotated to the left.	Page's defa
scale	Page scale.	`1`
width	Page width. If neither `height` nor `width` are defined, page will be rendered at the size defined in PDF. If you define `width` and `height` at the same time, `height` will be ignored. If you define `width` and `scale` at the same time, the width will be multiplied by a given factor.	Page's defa

Outline

Displays an outline (table of contents). Should be placed inside `<Document />`. Alternatively, it can have `pdf` prop passed, which can be obtained from `<Document />`'s `onLoadSuccess` callback function.

Props

Prop name	Description	Default value	Example values
className	Class name(s) that will be added to rendered element along with the default `react-pdf__Outline`.	n/a	• String: `"custom-class-name-1 custom-class-name-2"` • Array of strings: `["custom-class-name-1", "custom-class-name-2"]`
inputRef	A prop that behaves like ref, but it's passed to main `<div>` rendered by `<Outline>` component.	n/a	• Function: `(ref) => { this.myOutline = ref; }` • Ref created using `createRef`: `this.ref = createRef();` … `inputRef={this.ref}` • Ref created using `useRef`:

Prop name	Description	Default value	Example values

			`const ref = useRef();`
			`…`
			`inputRef={ref}`

onItemClick	Function called when an outline item has been clicked. Usually, you would like to use this callback to move the user wherever they requested to.	n/a	`({ dest, pageIndex, pageNumber }) => alert('Clicked an item from page ' + pageNumber + '!')`
onLoadError	Function called in case of an error while retrieving the outline.	n/a	`(error) => alert('Error while retrieving the outline! ' + error.message)`
onLoadSuccess	Function called when the outline is successfully retrieved.	n/a	`(outline) => alert('The outline has been successfully retrieved.')`

Thumbnail

Displays a thumbnail of a page. Does not render the annotation layer or the text layer. Does not register itself as a link target, so the user will not be scrolled to a Thumbnail component when clicked on an internal link (e.g. in Table of Contents). When clicked, attempts to navigate to the page clicked (similarly to a link in Outline). Should be placed inside `<Document />`. Alternatively, it can have `pdf` prop passed, which can be obtained from `<Document />`'s `onLoadSuccess` callback function.

Props

Props are the same as in `<Page />` component, but certain annotation layer and text layer-related props are not available:

- customTextRenderer
- onGetAnnotationsError
- onGetAnnotationsSuccess
- onGetTextError
- onGetTextSuccess
- onRenderAnnotationLayerError
- onRenderAnnotationLayerSuccess
- onRenderTextLayerError
- onRenderTextLayerSuccess
- renderAnnotationLayer
- renderForms
- renderTextLayer

On top of that, additional props are available:

Prop name	Description	Default value	Example values
className	Class name(s) that will be added to rendered element along with the default `react-pdf__Thumbnail`.	n/a	• String: `"custom-class-name-1 custom-class-name-2"` • Array of strings: `["custom-class-name-1", "custom-class-name-2"]`
onItemClick	Function called when a thumbnail has been	n/a	`({ dest, pageIndex,`

Prop name	Description	Default value	Example values
	clicked. Usually, you would like to use this callback to move the user wherever they requested to.		`pageNumber }) => alert('Clicked an item from page ' + pageNumber + '!')`

Useful links

- React-PDF Wiki

License

The MIT License.

Author

	Wojciech Maj

Thank you

This project wouldn't be possible without the awesome work of **Niklas Närhinen** who created its original version and without Mozilla, author of **pdf.js**. Thank you!

Sponsors

Thank you to all our sponsors! **Become a sponsor** and get your image on our README on GitHub.

Become a Sponsor

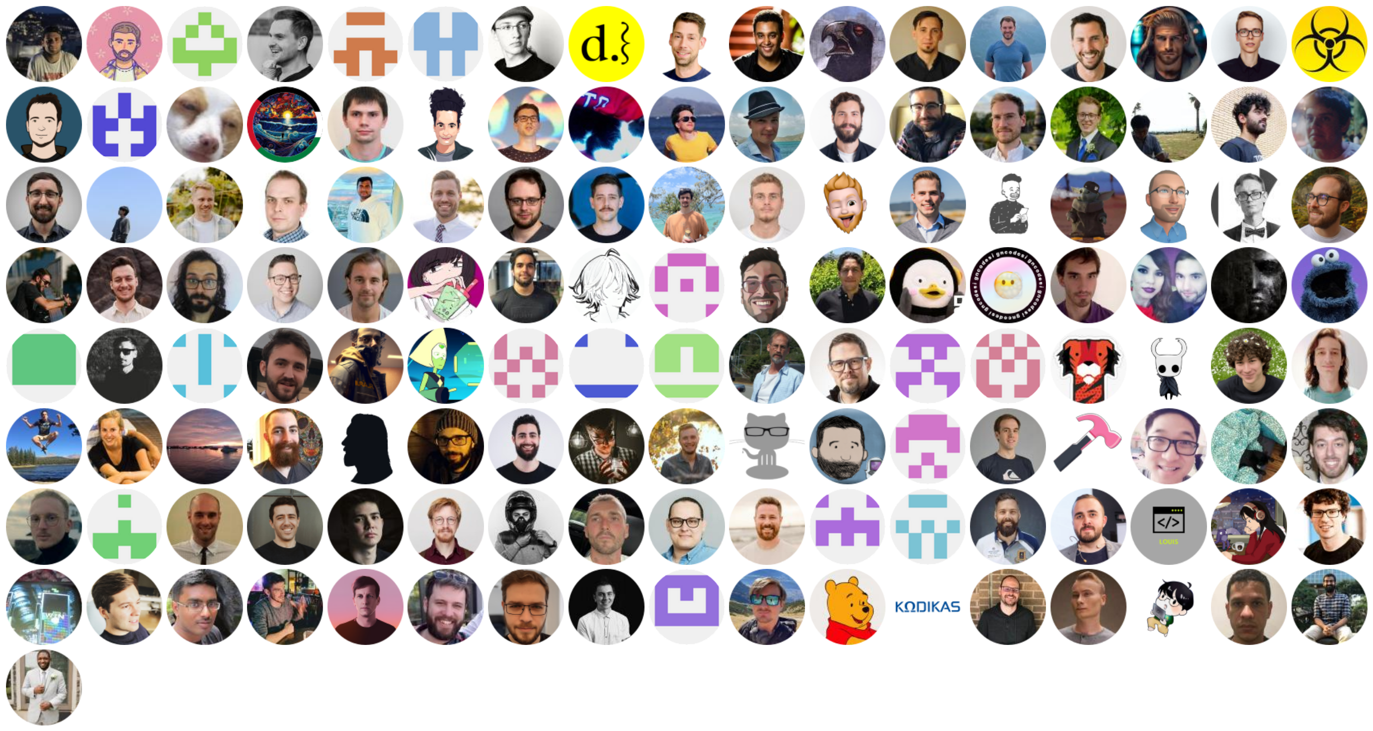
Backers

Thank you to all our backers! **Become a backer** and get your image on our README on GitHub.

Become a **Backer**

Top Contributors

Thank you to all our contributors that helped on this project!



Keywords

pdf pdf-viewer react

Provenance

Built and signed on

GitHub Actions

View build summary

Source Commit
github.com/wojtekmaj/react-pdf@5e0d135

Build File
.github/workflows/publish.yml

Public Ledger
Transparency log entry

Share feedback

Install

```
npm i react-pdf
```

Repository

github.com/wojtekmaj/react-pdf

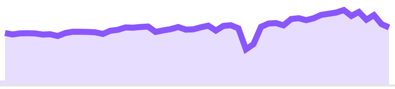
Homepage

github.com/wojtekmaj/react-pdf#readme

Fund this package

Weekly Downloads

1,060,945



Version

9.2.1

License

MIT

Unpacked Size

552 kB

Total Files

167

Last publish

5 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