react-pdf DT

5.6.0 • Public • Published 4 days ago

Readme

Explore BETA

8 Dependencies

384 Dependents

97 Versions

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 5.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:

- v4.x
- v3.x
- v2.x
- v1.x

Getting started

Compatibility

React

To use the latest version of React-PDF, your project needs to use React 16.3 or later.

If you use an older version of React, please refer to the table below to a find suitable React-PDF version. Don't worry - as long as you're running React 15.5 or later, you won't be missing out a lot!

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

Internet Explorer

Internet Explorer is not supported in React-PDF 5.x or later. If you need Internet Explorer support, you will need to use React-PDF 4.x instead. Don't worry - it still receives bug fixes and even occasional small features!

Installation

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

Usage

Here's an example of basic usage:

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

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

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

  return (
    <div>
      <Document
        file="somefile.pdf"
        onLoadSuccess={onDocumentLoadSuccess}
      >
        <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**.

Enable PDF.js worker

It is crucial for performance to use PDF.js worker whenever possible. This ensures that PDF files will be rendered in a separate thread without affecting page performance. To make things a little easier, we've prepared several entry points you can use.

Webpack

Instead of directly importing modules you need from `'react-pdf'`, import them like so:

```
// using ES6 modules
import { Document, Page } from 'react-pdf/dist/esm/entry.webpack';

// using CommonJS modules
import { Document, Page } from 'react-pdf/dist/umd/entry.webpack';
```

Parcel

Instead of directly importing modules you need from `'react-pdf'`, import them like so:

```
// using ES6 modules
import { Document, Page } from 'react-pdf/dist/esm/entry.parcel';

// using CommonJS modules
import { Document, Page } from 'react-pdf/dist/umd/entry.parcel';
```

Create React App

Create React App uses Webpack under the hood, so you can follow **Webpack instructions**.

Standard instructions will also work. In Create React App, you can copy `pdf.worker.js` file from `pdfjs-dist/build` to `public` directory in order for it to be copied to your project's output folder at build time.

Standard (Browserify and others)

If you use Browserify or other bundling tools, you will have to make sure on your own that `pdf.worker.js` file from `pdfjs-dist/build` is copied to your project's output folder.

If you don't need to debug `pdf.worker.js`, you can use `pdf.worker.min.js` file instead, which is roughly half the size. For this to work, however, you will need to specify `workerSrc` manually like so:

```
import { pdfjs } from 'react-pdf';
pdfjs.GlobalWorkerOptions.workerSrc = 'pdf.worker.min.js';
```

Alternatively, you could use the minified `pdf.worker.min.js` from an external CDN:

```
import { pdfjs } from 'react-pdf';
pdfjs.GlobalWorkerOptions.workerSrc = `//cdnjs.cloudflare.com/ajax/libs/pdf.js/${pdfjs.version}/pdf.worker.m
```

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:

```
// using ES6 modules
import 'react-pdf/dist/esm/Page/AnnotationLayer.css';

// using CommonJS modules
import 'react-pdf/dist/umd/Page/AnnotationLayer.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 "cMapUrl" and "cMapPacked" API paramete
```

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`.

Webpack

Add `copy-webpack-plugin` to your project if you haven't already:

```
npm install copy-webpack-plugin --save-dev
```

Now, in your Webpack config, import the plugin:

```
import CopyWebpackPlugin from 'copy-webpack-plugin';
```

and in `plugins` section of your config, add the following:

```
new CopyWebpackPlugin([
  {
    from: 'node_modules/pdfjs-dist/cmaps/',
    to: 'cmaps/'
  },
]),
```

Parcel, Browserify and others

If you use Parcel, Browserify or other bundling tools, you will have to make sure on your own that cMaps are copied to your project's output folder.

Setting up React-PDF

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

```
<Document
  options={{
    cMapUrl: 'cmaps/',
    cMapPacked: true,
  }}
/>
```

Alternatively, you could use cMaps from external CDN:

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

<Document
  options={{
    cMapUrl: `//cdn.jsdelivr.net/npm/pdfjs-dist@${pdfjs.version}/cmaps/`,
    cMapPacked: true,
  }}
/>
```

User guide

Document

Loads a document passed using `file` prop.

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__Document`.	n/a	• String: `"custom-class-name-1 custom-class-name-2"` • Array of strings: `["custom-class-name-1", "custom-class-name-2"]`
error	What the component should display in case of an error.	`"Failed to load PDF file."`	• String: `"An error occurred!"` • React element: `<div>An error occurred!</div>` • Function: `this.renderError`

Prop name	Description	Default value	Example values
externalLinkTarget	Link target for external links rendered in annotations.	unset, which means that default behavior will be used	One of valid values for `target` attribute. • `"_self"` • `"_blank"` • `"_parent"` • `"_top"`

Prop name	Description	Default value	Example values
file	What PDF should be displayed. Its value can be an URL, a file (imported using `import ... from ...` or from file input form element), or an object with parameters (`url` - URL; `data` - data, preferably Uint8Array; `range` - PDFDataRangeTransport; `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`). **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.	n/a	• URL: `"http://example.com/sample.pdf"` • File: `import sample from '../static/sample.pdf'` and then `sample` • Parameter object: `{ url: 'http://example.com/sample.pdf', httpHeaders: { 'X-CustomHeader': '40359820958024350238508234' }, withCredentials: true }`

Prop name	Description	Default value	Example values
imageResourcesPath	The path used to prefix the src attributes of annotation SVGs.	n/a (pdf.js will fallback to an empty string)	`"/public/images/"`
inputRef	A prop that behaves like ref, but it's passed to main `<div>` rendered by `<Document>` component.	n/a	• Function: `(ref) => { this.myDocument = ref; }` • Ref created using `React.createRef`: `this.ref = React.createRef();` ... `inputRef={this.ref}` • Ref created using `React.useRef`: `const ref = React.useRef();` ... `inputRef={ref}`
loading	What the component should display while loading.	`"Loading PDF…"`	• String: `"Please wait!"` • React element: `<div>Please wait!</div>` • Function: `this.renderLoader`

Prop name	Description	Default value	Example values
noData	What the component should display in case of no data.	`"No PDF file specified."`	• String: `"Please select a file."` • React element: `<div>Please select a file.</div>` • Function: `this.renderNoData`
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	`({ pageNumber }) => alert('Clicked an item from page ' + pageNumber + '!')`
onLoadError	Function called in case of an error while loading a document.	n/a	`(error) => alert('Error while loading document! ' + error.message)`
onLoadProgress	Function called, potentially multiple times, as the loading progresses.	n/a	`({ loaded, total }) => alert('Loading a document: ' + (loaded / total) * 100 + '%');`
onLoadSuccess	Function called when the document is successfully loaded.	n/a	`(pdf) => alert('Loaded a file with ' + pdf.numPages + ' pages!')`

Prop name	Description	Default value	Example values
onPassword	Function called when a password-protected PDF is loaded.	A function that prompts the user for password	`(callback) => callback('s3cr3t_p4ssw0rd')`
onSourceError	Function called in case of an error while retrieving document source from `file` prop.	n/a	`(error) => alert('Error while retrieving document source! ' + error.message)`
onSourceSuccess	Function called when document source is successfully retrieved from `file` prop.	n/a	`() => alert('Document source retrieved!')`
options	An object in which additional parameters to be passed to PDF.js can be defined. For a full list of possible parameters, check PDF.js documentation on DocumentInitParameters.	n/a	`{ cMapUrl: 'cmaps/', cMapPacked: true }`
renderMode	Rendering mode of the document. Can be `"canvas"`, `"svg"` or `"none"`.	`"canvas"`	`"svg"`

Prop name	Description	Default value	Example values
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 linking between pages inside a document may not be working correctly.

Props

Prop name	Description	Default value	Example values
canvasBackground	Canvas background color. Any valid `canvas.fillStyle` can be used. If you set `renderMode` to `"svg"` this prop will be ignored.	n/a	`"transparent"`

Prop name	Description	Default value	Example values
canvasRef	A prop that behaves like ref, but it's passed to `<canvas>` rendered by `<PageCanvas>` component. If you set `renderMode` to `"svg"` this prop will be ignored.	n/a	• Function: `(ref) => { this.myPage = ref; }` • Ref created using `React.createRef`: `this.ref = React.createRef();` ... `inputRef={this.ref}` • Ref created using `React.useRef`: `const ref = React.useRef();` ... `inputRef={ref}`

Prop name	Description	Default value	Example values
className	Class name(s) that will be added to rendered element along with the default `react-pdf__Page`.	n/a	• String: `"custom-class-name-1 custom-class-name-2"` • Array of strings: `["custom-class-name-1", "custom-class-name-2"]`
customTextRenderer	A function that customizes how a text layer is rendered. Passes itext item and index for item.	n/a	`({ str, itemIndex }) => { return (<mark>{str}</mark>) }`
error	What the component should display in case of an error.	`"Failed to load the page."`	• String: `"An error occurred!"` • React element: `<div>An error occurred!</div>` • Function: `this.renderError`

Prop name	Description	Default value	Example values
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 default height	`300`
imageResourcesPath	The path used to prefix the src attributes of annotation SVGs.	n/a (pdf.js will fallback to an empty string)	`"/public/images/"`

Prop name	Description	Default value	Example values
inputRef	A prop that behaves like ref, but it's passed to main `<div>` rendered by `<Page>` component.	n/a	• Function: `(ref) => { this.myPage = ref; }` • Ref created using `React.createRef`: `this.ref = React.createRef(); ... inputRef={this.ref}` • Ref created using `React.useRef`: `const ref = React.useRef(); ... inputRef={ref}`

Prop name	Description	Default value	Example values
loading	What the component should display while loading.	`"Loading page..."`	• String: `"Please wait!"` • React element: `<div>Please wait!</div>` • Function: `this.renderLoader`
noData	What the component should display in case of no data.	`"No page specified."`	• String: `"Please select a page."` • React element: `<div>Please select a page.</div>` • Function: `this.renderNoData`
onLoadError	Function called in case of an error while loading the page.	n/a	`(error) => alert('Error while loading page! ' + error.message)`

Prop name	Description	Default value	Example values
onLoadSuccess	Function called when the page is successfully loaded.	n/a	`(page) => alert('Now displaying a page number ' + page.pageNumber + '!')`
onRenderError	Function called in case of an error while rendering the page.	n/a	`(error) => alert('Error while loading page! ' + error.message)`
onRenderSuccess	Function called when the page is successfully rendered on the screen.	n/a	`() => alert('Rendered the page!')`
onGetAnnotationsSuccess	Function called when annotations are successfully loaded.	n/a	`(annotations) => alert('Now displaying ' + annotations.length + ' annotations!')`
onGetAnnotationsError	Function called in case of an error while loading annotations.	n/a	`(error) => alert('Error while loading annotations! ' + error.message)`

Prop name	Description	Default value	Example values
onGetTextSuccess	Function called when text layer items are successfully loaded.	n/a	`(items) => alert('Now displaying ' + items.length + ' text layer items!')`
onGetTextError	Function called in case of an error while loading text layer items.	n/a	`(error) => alert('Error while loading text layer items! ' + error.message)`
pageIndex	Which page from PDF file should be displayed, by page index.	`0`	`1`
pageNumber	Which page from PDF file should be displayed, by page number. If provided, `pageIndex` prop will be ignored.	`1`	`2`
renderAnnotationLayer	Whether annotations (e.g. links) should be rendered.	`true`	`false`
renderInteractiveForms	Whether interactive forms should be rendered. `renderAnnotationLayer` prop must be set to `true`.	`false`	`true`

Prop name	Description	Default value	Example values
renderMode	Rendering mode of the document. Can be `"canvas"`, `"svg"` or `"none"`.	`"canvas"`	`"svg"`
renderTextLayer	Whether a text layer should be rendered.	`true`	`false`
rotate	Rotation of the page in degrees. `90` = rotated to the right, `180` = upside down, `270` = rotated to the left.	Page's default setting, usually `0`	`90`
scale	Page scale.	`1.0`	`0.5`
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 default width	`300`

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

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 `React.createRef`: `this.ref = React.createRef();` … `inputRef={this.ref}` • Ref created using `React.useRef`: `const ref = React.useRef();` … `inputRef={ref}`

Prop name	Description	Default value	Example values
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	`({ 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.')`

Useful links

- React-PDF Wiki

License

The MIT License.

Author

	Wojciech Maj

kontakt@wojtekmaj.pl
https://wojtekmaj.pl

Thank you

This project wouldn't be possible without awesome work of Niklas Närhinen **niklas@narhinen.net** who created its initial 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.

Backers

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

Top Contributors

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

Install

```
> npm i react-pdf
```

Repository

github.com/wojtekmaj/react-pdf

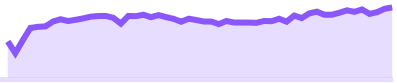
Homepage

github.com/wojtekmaj/react-pdf#readme

❤ **Fund** this package

Weekly Downloads

368,089



Version

5.6.0

License

MIT

Keywords

pdf pdf-viewer react

Unpacked Size

406 kB

Total Files

90

Issues

93

Pull Requests

8

Last publish

4 days ago

Collaborators

Try on RunKit

Report malware

Support

Help

Community

Advisories

Status

Contact npm

Company

About

Blog

Press

Terms & Policies

Policies

Terms of Use

Code of Conduct

Privacy