You may extend the list of supported browsers by providing additional polyfills (e.g. for `Array.prototype.at` or `Promise.allSettled`) 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 v5.

If you need to support Internet Explorer 11, you will need to use React-PDF v4.

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` .

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.

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.js',
  import.meta.url,
).toString();
```

> ⓘ 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.js` 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.j
const pdfWorkerPath = path.join(pdfjsDistPath, 'build', 'pdf.worker.js')
```

```
fs.copyFileSync(pdfWorkerPath, './dist/pdf.worker.js');
```

Use external CDN

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

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

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.js',
+  'pdfjs-dist/legacy/build/pdf.worker.min.js',
   import.meta.url,
 ).toString();
```

or:

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

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/esm/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/esm/Page/TextLayer.css';
```