


[< BACK](#)

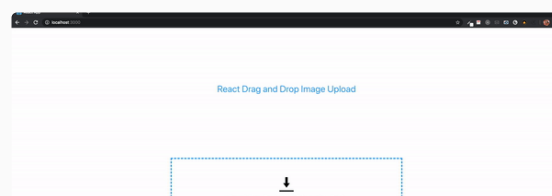
Create a Drag-and-Drop Zone in React with react-dropzone

 **Ejiro Thankgod**, April 19th, 2022 · 4 min read



Drag and drop is a platform whereby an application uses drag-and-drop features on browsers. The user selects a draggable item with a mouse or touchpad, drags those files to a droppable element (drop zone), and drops them by releasing the mouse button.

[React-dropzone](#) is a set of React libraries to help you build complex drag-and-drop interfaces while keeping your components destructured. The most common use cases for drag-and-drop in React comprise uploading files, rearranging images/files, and moving files between multiple folders. We'll create a simple React application with a drag-and-drop interface, just like the one below.



By the end of this tutorial, in our React application we'll be able to do the following:

- Drag-and-drop a file on the drag-and-drop zone,
- Click event which initiates file selection dialog
- Display file preview on the drop zone,
- List file name, file type, and file size,
- Alternative option 'click to open' button,
- Indicate the file received on the drop zone,

Getting Started

First of all, we are going to create our React app. I believe we already know this process, but we will start from scratch here, for starters. We are creating our React app from our terminal using `npx create-react-app drag-n-drop` for those using npm, while others using yarn should run `yarn create React-app drag-n-drop`.

After, run `npm start` or `yarn start` to begin. You can manually delete some of the generated boilerplate files and code, such as:

- `favicon.ico`
- all logo files (`logo192.png`, `logo512.png`, `logo.svg`)
- `manifest.json`
- `robots.txt`
- `reportWebVitals.js`
- `setupTests.js`
- `App.test.js`

Alternatively, you could run `npx clean-React-app` to remove such files.

How to Implement a Drag and Drop Zone in React

We won't be building all the logic and components from scratch. Instead, we'll use one of the most common React drag-and-drop libraries available: React-dropzone. React-dropzone is a very powerful library and custom component. It's a simple React hook to create an HTML-5 compliant drag-and-drop zone for files. Dropzone provides additional functions such as customizing the dropzone, restricting file types, etc. If you need a visual explanation of a drag-and-drop with React-dropzone, you can watch [this video](#).

We will install our React dropzone with `npm install`

React-dropzone or yarn add React-dropzone. Right now, our code should be looking something like this in our App.js component.

```
1 import React from "react";
2
3 function App() {
4   return <div></div>;
5 }
6
7 export default App;
```

We want to create an area where, when we click on it, the React-dropzone library initiates the file selection dialog, allowing us to select and upload files. We'll start by creating a component.

```
1  /*Dropzone.js*/
2
3  import React from "React";
4  import { useDropzone } from "React-dropzone";
5
6  function Dropzone({ open }) {
7    const { getRootProps, getInputProps } = useDropzone({});
8    return (
9      <div {...getRootProps({ className: "dropzone" })}>
10        <input className="input-zone" {...getInputProps()} />
11        <div className="text-center">
12          <p className="dropzone-content">
13            Drag'n'drop some files here, or click to select files
14          </p>
15        </div>
16      </div>
17    );
18  }
19
20
21 export default Dropzone;
```

Now we have to import the Dropzone to our App.js ;

```
1  /*App.js*/
2
3  import React from "React";
4  import Dropzone from "../Dropzone";
5
6  function App() {
7    // const [images, setImages] = useState({});
8    return (
9      <div>
10        <div className="container">
11          <h1 className="text-center">Drag and Drop Test</h1>
12          <Dropzone />
13        </div>
14      </div>
15    );
16  }
17
18 export default App;
```

Now that we have done that, our browser should look like this.

Styling

Although our React-dropzone is working well, it doesn't look pleasant, so we will style it a little bit so that it may look more

pleasant, so we will style it a little bit so that it may look more like an actual dropzone. Now we move to `index.css` to implement some styling.

```
1  /**index.css*/
2  body {
3    text-align: center;
4    padding: 20px;
5    border: 3px blue dashed;
6    width: 60%;
7    margin: auto;
8  }
```

Copy

We must import `index.css` in our `App.js`. The results are as follows.

As you can see, this is our React-dropzone. However, it still needs more styling. I want to add some components to it to make it look more functional because, right now, the only thing it can do is the drag-and-drop and the click event, which initiate the file selection dialog. We also need to have a path to store files drawn into the dropzone.

Accepting Files and Recording File Properties

If you're following this guide, you will have noticed that the React-dropzone does not accept files nor list the file name, file type, and file size; that's what we'll be working on. We have to create a function for our dropzone to enable it to accept files and display file names, types, and sizes. Our code should look like this.

```
1  import React from "React";
2  import { useDropzone } from "React-dropzone";
3  import "../index.css";
4
5  function Dropzone({ open }) {
6    const { getRootProps, getInputProps, acceptedFiles } =
7      useDropzone({});
8
9    const files = acceptedFiles.map((file) => {
10      <li key={file.path}>
11        {file.path} - {file.size} bytes
12      </li>
13    });
14
15    return (
16      <div className="container">
17        <div {...getRootProps({ className: "dropzone" })}>
18          <input {...getInputProps()} />
19          <p>Drag 'n' drop some files here</p>
20        </div>
21        <aside>
22          <ul>{files}</ul>
23        </aside>
24      </div>
25    );
26  }
27
28  export default Dropzone;
```

Copy

With our app looking like this;

When we drag a file into our dropzone, the file is accepted, and

its properties (name, type, and size) are displayed.

Adding our Buttons and Styling them

Our React-dropzone is getting into shape, but we would like to manually add a button to initiate the file selection dialog. All we have to do is create a button component beneath our `<p>` in our `Dropzone.js`.

```
1  ...
2  <p className="dropzone-content">
3    Drag' n' drop some files here, or click to select files
4  </p>
5  })
6  <button type="button" onClick={open} className="btn">
7    Click to select files
8  </button>
9  ...
```

Copy

All we have to do now is style our button with the class name `btn` and make it look beautiful. We'll go to our `index.css` and beautify our buttons.

```
1  .btn {
2    border: none;
3    text-align: center;
4    background-color: rgb(218, 216, 216);
5    height: 50px;
6    border-radius: 12px;
7    color: black;
8    font-weight: bold;
9    transition-duration: 0.6s;
10 }
11
12 .btn:hover {
13   background-color: blue;
14   color: aliceblue;
15 }
```

Copy

And it looks nice! Up next, we'll be working on the `isDragActive` component. It is set to detect if a draggable file is within the dropzone; then the text at the dropzone changes from 'drag'n'drop some files here or click to select' to 'release to drop the file here'.

Implementing the `isDragActive` Components

We'll need some code.

```
1  /**Dropzone.js*/
2
3  ...
4
5  function Dropzone({ open }) {
6    const { getRootProps, getInputProps, isDragActive, acceptedFiles } =
7      useDropzone({});
8
9    ...
10
11   return (
12     <div {...getRootProps({ className: "dropzone" })}>
13       <input className="input-zone" {...getInputProps()} />
14       <div className="text-center">
15         {isDragActive ? (
```

Copy

```

15   (isDragActive ? (
16     <p className="dropzone-content">
17       Release to drop the files here
18     </p>
19   ) : (
20     <p className="dropzone-content">
21       Drag'n'drop some files here, or click to select files
22     </p>
23   ))
24   <button type="button" onClick={open} className="btn">
25     Click to select files
26   </button>
27 </div>
28 <aside>
29   <ul>{files}</ul>
30 </aside>
31 </div>
32 );
33 }
34
35 export default Dropzone;

```

Open Source Session Replay

OpenReplay is an open-source, session replay suite that lets you see what users do on your web app, helping you troubleshoot issues faster. OpenReplay is self-hosted for full control over your data.

Start enjoying your debugging experience - [start using OpenReplay for free](#).

Display Image Preview

To show images in grid layout (another cool feature) we'll create a component called `imagegrid.js`. We will use another library called [React-dnd](#).

```

1  /**ImageGrid.js**/
2
3  ...
4
5  function App() {
6    const [images, setImages] = useState([]);
7    const onDrop = useCallback((acceptedFiles) => {
8      acceptedFiles.map((file) => {
9        const reader = new FileReader();
10
11        reader.onload = function (e) {
12          setImages((prevState) => [
13            ...prevState,
14            { id: cuid(), src: e.target.result },
15          ]);
16        };
17
18        reader.readAsDataURL(file);
19        return file;
20      });
21    }, []);
22
23    return (
24      <main className="App">
25        <h1 className="text-center">Drag and Drop Test</h1>
26        <Dropzone onDrop={onDrop} accept={"image/*"} />
27        <ImageGrid images={images} />
28      </main>
29    );
30  }
31
32  export default App;

```

Copy

If you check `<Dropzone onDrop={onDrop} accept=`
`["image/*"] />` you will see that we made it to only accept

`{ image/ ~ } //>`, you will see that we made it to only accept images, but you can allow multiple file types. We will also apply some styling to it.

```
1  /**index.css*/
2
3  .file-list {
4    display: flex;
5    flex-wrap: wrap;
6    width: 65%;
7    margin: 20px auto;
8    padding: 10px;
9    border: 3px dotted black;
10 }
11
12 .file-list img {
13   height: 300px;
14   width: 300px;
15   object-fit: cover;
16 }
```

Copy

And our drag-and-drop zone is working perfectly with our well-styled button for manually initiating file dialog and our display preview on our images.

Conclusion

There are different React drag-and-drop libraries ranging from React-beautiful-dnd down to React-grid-layout and, finally, React-dnd (which I used for our image preview, and I'll be discussing it in my next article). Although they have their pros and cons, they all work uniquely depending on what you use them for. For example, we used React-dropzone to customize our drag-and-drop zone earlier, and it looked winsome and responsive. There are so many ways you could customize your React-dropzone. Maybe you want a specific file type as we did there, or you want an accept-or-reject list? You can also implement that and open the file dialog programmatically with a button. You could also check [their website](#) for more information.

For reference, here's the complete code for the project here.

```
1  /**App.js*/
2
3  import React, { useCallbck, useState } from "React";
4  import cuid from "cuid";
5  import Dropzone from "../Dropzone";
6  import ImageGrid from "../components/ImageGrid";
7  import "../index.css";
8  import "../App.css";
9
10 function App() {
11   const [images, setImages] = useState([]);
12   const onDrop = useCallback((acceptedFiles) => {
13     acceptedFiles.map((file) => {
14       const reader = new FileReader();
15
16       reader.onload = function (e) {
17         setImages((prevState) => [
18           ...prevState,
19           { id: cuid(), src: e.target.result },
20         ]);
21       };
22     });
23     reader.readAsDataURL(file);
24     return file;
25   });
26   }, []);
27
28   return (
```

Copy

```

29   <main className="App">
30     <h1 className="text-center">Drag and Drop Test</h1>
31     <Dropzone onDrop={onDrop} accept={"image/*"} />
32
33     <ImageGrid images={images} />
34   </main>
35 );
36 }
37 export default App;

```

```

1  /**ImageGrid.js*/
2
3  import React from "React";
4  // Rendering individual images
5  const Image = ({ image }) => {
6    return (
7      <div className="file-item">
8        <img
9          alt={`img - ${image.id}`}
10         src={image.src}
11         className="file-img"
12       />
13     </div>
14   );
15 };
16
17 // ImageList Component//
18 const ImageGrid = ({ images }) => {
19   // render each image by calling Image component
20   const renderImage = (image, index) => {
21     return <Image image={image} key={`-${image.id}-image`} />;
22   };
23   // Return the list of files//
24   return (
25     <section className="file-list">{images.map(renderImage)}</section>
26   );
27 };
28
29 export default ImageGrid;

```

```

1  /**Dropzone.js*/
2
3  import React from "React";
4  import { useDropzone } from "React-dropzone";
5
6  function Dropzone({ onDrop, accept, open }) {
7    const { getRootProps, getInputProps, isDragActive, acceptedFiles } =
8      useDropzone({
9        accept,
10        onDrop,
11      });
12
13    const files = acceptedFiles.map((file) => (
14      <li key={file.path}>
15        {file.path} - {file.size} bytes
16      </li>
17    ));
18
19    return (
20      <div>
21        <div {...getRootProps({ className: "dropzone" })}>
22          <input className="input-zone" {...getInputProps()} />
23          <div className="text-center">
24            {isDragActive ? (
25              <p className="dropzone-content">
26                Release to drop the files here
27              </p>
28            ) : (
29              <p className="dropzone-content">
30                Drag' n' drop some files here, or click to select files
31              </p>
32            )}
33            <button type="button" onClick={open} className="btn">
34              Click to select files
35            </button>
36          </div>
37        </div>
38        re
39        <aside>
40          <ul>{files}</ul>
41        </aside>
42      </div>
43    );
44  }
45
46 export default Dropzone;

```



```
1  /**index.css*/
2  body {
3    text-align: center;
4  }
5
6  .dropzone {
7    text-align: center;
8    padding: 20px;
9    border: 3px blue dashed;
10   width: 60%;
11   margin: auto;
12 }
13
14 .btn {
15   border: none;
16   text-align: center;
17   background-color: rgb(218, 216, 216);
18   height: 50px;
19   border-radius: 12px;
20   color: black;
21   font-weight: bold;
22   transition-duration: 0.6s;
23 }
24
25 .btn:hover {
26   background-color: blue;
27   color: aliceblue;
28 }
29
30 .file-list {
31   /* border: 3px dotted black; */
32
33   display: flex !important;
34   flex-wrap: wrap;
35   width: auto;
36   padding: 10px 20px;
37   margin: 20px 30px;
38   /* border: 3px dotted black; */
39 }
40
41 .file-list img {
42   height: 100%;
43   width: 100px;
44   padding-right: 10px;
45   object-fit: cover;
```

Copy 

We've successfully built a small project for dragging and dropping files. There are many more functionalities in React. For more, check out the links below.

- [Documentation and examples](#)
- [Github source code](#)
- [React.js tutorial-dropzone](#)

Happy coding!

More articles from OpenReplay Blog





React State Management Using Easy Peasy

A simple way to manage state in React apps

April 18th, 2022 · 10 min read

Ajax Battle: XMLHttpRequest vs the Fetch API

Which Ajax API is best for your application? We examine the pros and cons of the ancient XMLHttpRequest and its modern Fetch equivalent.

April 17th, 2022 · 5 min read