useUploadFile

Overview

The `useUploadFile` hook is a versatile utility for uploading files to a server, supporting both browser-based `File` objects and React Native `uri`-based file representations. Unlike basic file upload utilities, this hook is designed to handle diverse environments seamlessly while maintaining flexibility and ease of integration. It also allows for custom path structures to organize uploaded files efficiently.

Usage Example

```
import React, { useState } from "react";
import useUploadFile from "@replyke/react-js";

function FileUploader() {
  const uploadFile = useUploadFile();
  const [uploadStatus, setUploadStatus] = useState<string>("");

  const handleFileUpload = async (file: File) => {
    try {
      setUploadStatus("Uploading...");

      const response = await uploadFile(file, ["user-data", "images"]);
      if (response) {
        setUploadStatus(`Upload successful! File URL: ${response.publicPath}`);
      }
    } catch (error) {
      setUploadStatus(`Upload failed: ${error.message}`);
    }
  };

  return (
    <div>
      <input
        type="file"
        onChange={(e) => {
          if (e.target.files?.[0]) {
            handleFileUpload(e.target.files[0]);
          }
        }}
```

```
        />
        <p>{uploadStatus}</p>
      </div>
    );
}
```

Parameters & Returns

Parameters

The hook returns a single function, `uploadFile`, which accepts the following arguments:

Parameter	Type	Required	Description
`file`	`UniversalFile`	Yes	The file to be uploaded. Can be a browser `File` or a React Native `{ uri, name, type }` object.
`pathParts`	`string[]`	Yes	An array of strings representing the hierarchical path to store the file.

UniversalFile Type

The `file` parameter can be one of the following:

- **BrowserFile**: A native browser `File` object.
- **RNFile**: An object with the following properties:
  - `uri`: The URI of the file in the React Native environment.
  - `name`: The name of the file.
  - `type`: (Optional) The MIME type of the file. Defaults to `application/octet-stream`.

Returns

The `uploadFile` function returns a promise resolving to an object of type `UploadResponse` or throws an error:

Return Value	Type	Description
`fileId`	`string`	A unique identifier for the uploaded file.
`relativePath`	`string`	The server-stored relative path of the uploaded file.
`publicPath`	`string`	A public URL to access the uploaded file.

Features

1. **Environment-Agnostic**: Supports both browser and React Native file uploads.
2. **Dynamic Path Management**: Allows the user to define custom storage paths for organizing files.
3. **Error Handling**: Throws descriptive errors for invalid arguments or failed uploads.
4. **Multipart Form Support**: Automatically constructs the correct `FormData` structure based on the file type.

Best Practices

1. **Validate Input**: Ensure that the `file` and `pathParts` parameters are correctly formatted before invoking the hook.
2. **Use Path Hierarchies**: Leverage the `pathParts` parameter to organize files effectively in the storage system.
3. **Handle Errors Gracefully**: Implement user-friendly error messages and fallback mechanisms in case of upload failures.

Last updated on May 6, 2025