File Uploads in Express:

A large number of mobile apps and websites allow users to upload profile pictures and other files. Therefore, handling files upload is a common requirement while building a REST API with Node.js & Express.

How to handle single and multiple files uploads with Node.js and Express backend and save uploaded files on the server.

express-fileupload

Simple express middleware for uploading files.

Install

```
# With NPM

npm i express-fileupload
```

When you upload a file, the file will be accessible from req.files.

req.files is used to handle file uploads. It is similar to req.body and is turned on either by express.bodyParser() or express.multipart() middlewares. Express.js (and other modules behind the scene) process the request data (which isusually a form) and give us extensive information in the req.files.FIELD_NAME object.

Example: You're uploading a file called car.jpg

Your input's name field is foo: <input name="foo" type="file" />

In your express server request, you can access your uploaded file from req.files.foo:

```
app.post('/upload', function(req, res) {
  console.log(req.files.foo); // the uploaded file object
});
```

The **req.files.foo** object will contain the following:

- req.files.foo.name: "car.jpg"
- req.files.foo.mv: A function to move the file elsewhere on your server. Can take a callback or return a promise.
- req.files.foo.mimetype: The mimetype of your file

- req.files.foo.data: A buffer representation of your file, returns empty buffer in case useTempFiles option was set to true.
- req.files.foo.tempFilePath: A path to the temporary file in case useTempFiles option was set to true.
- req.files.foo.truncated: A boolean that represents if the file is over the size limit
- req.files.foo.size: Uploaded size in bytes
- req.files.foo.md5: MD5 checksum of the uploaded file

For more information on express file uploads visit:

https://attacomsian.com/blog/uploading-files-nodejs-express