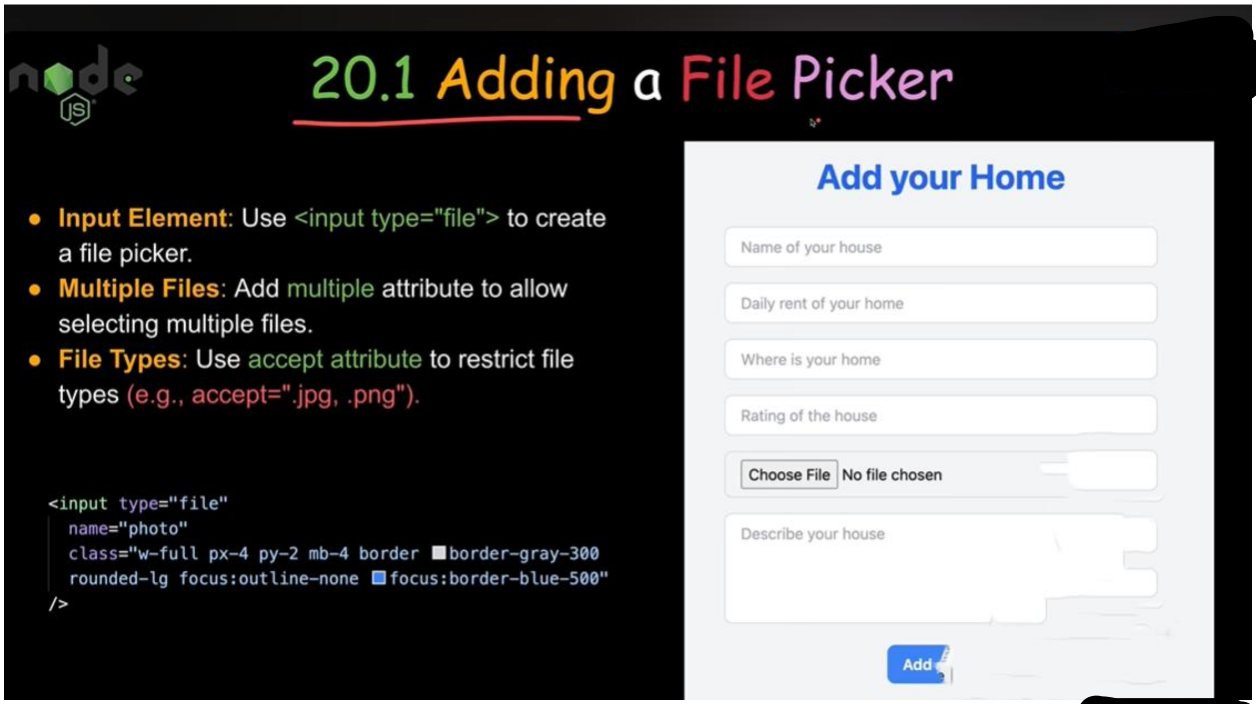
Files Upload and Download

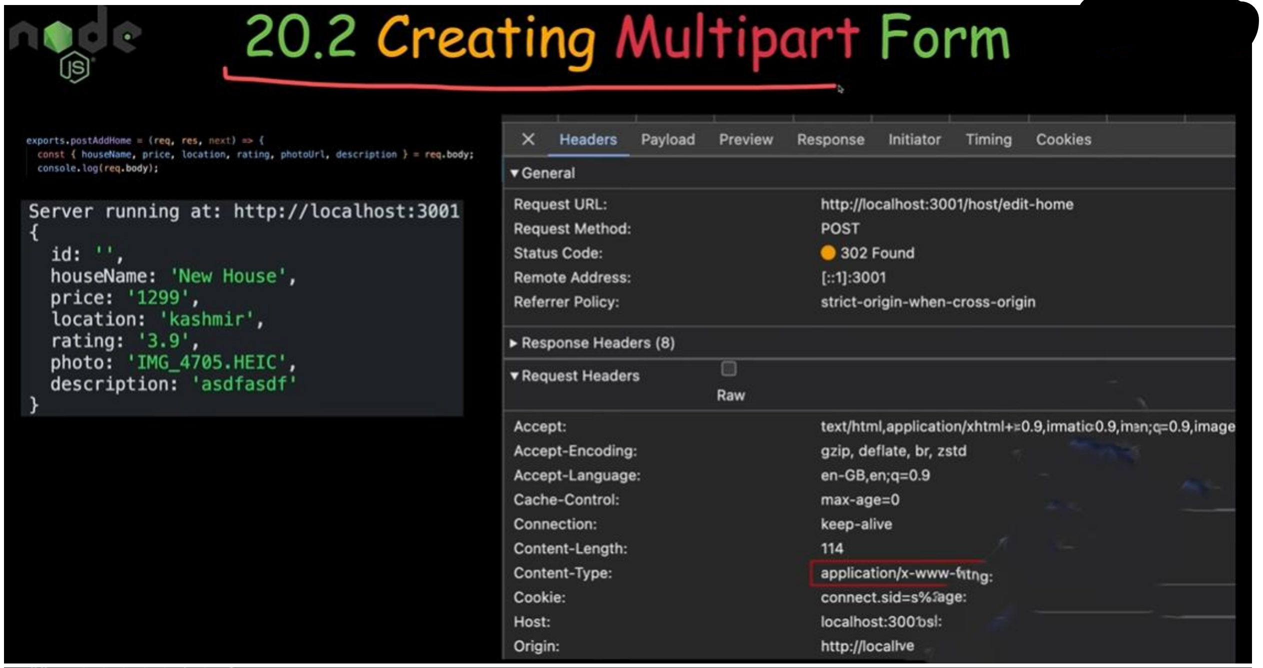


[redacted]

20.1 Adding a File Picker

- **Input Element**: Use <input type="file"> to create a file picker.
- **Multiple Files**: Add multiple attribute to allow selecting multiple files.
- **File Types**: Use accept attribute to restrict file types (e.g., accept=".jpg, .png").

```
<input type="file"
  name="photo"
  class="w-full px-4 py-2 mb-4 border border-gray-300
  rounded-lg focus:outline-none focus:border-blue-500"
/>
```



[redacted]

20.2 Creating Multipart Form

```
exports.postAddHome = (req, res, next) => {
  const { houseName, price, location, rating, photoUrl, description } = req.body;
  console.log(req.body);
```

```
Server running at: http://localhost:3001
{
  id: '',
  houseName: 'New House',
  price: '1299',
  location: 'kashmir',
  rating: '3.9',
  photo: 'IMG_4705.HEIC',
  description: 'asdfasdf'
}
```

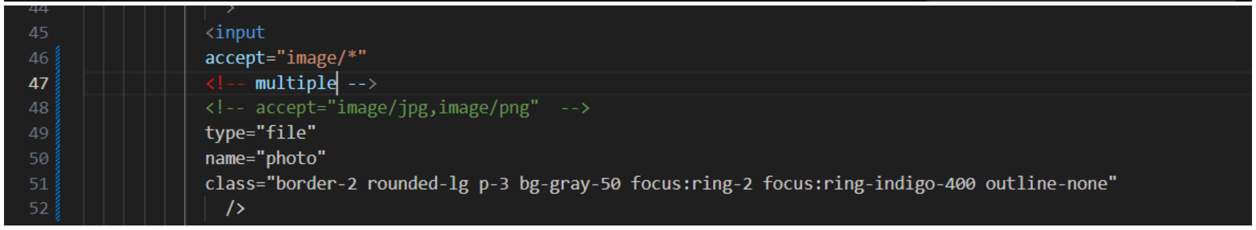
▼General

Request URL:	http://localhost:3001/host/edit-home
Request Method:	POST
Status Code:	● 302 Found
Remote Address:	[::1]:3001
Referrer Policy:	strict-origin-when-cross-origin

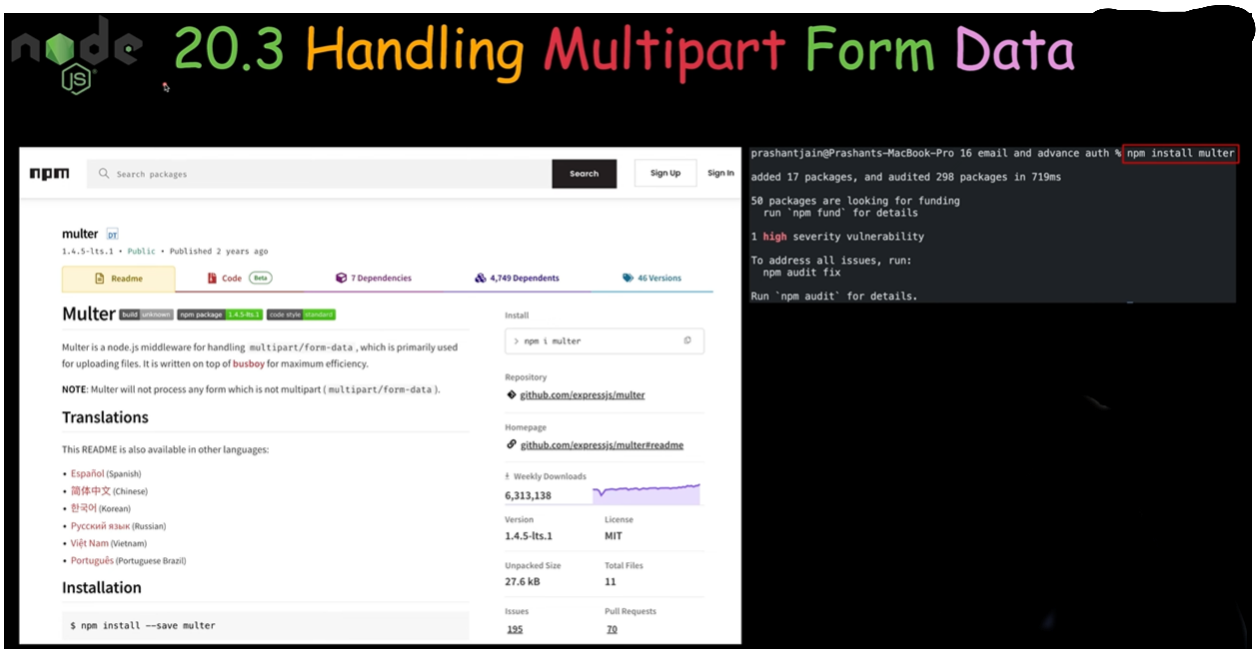
▶ Response Headers (8)

▼ Request Headers Raw

Accept:	text/html,application/xhtml+=0.9,imatic0.9,man;q=0.9,image
Accept-Encoding:	gzip, deflate, br, zstd
Accept-Language:	en-GB,en;q=0.9
Cache-Control:	max-age=0
Connection:	keep-alive
Content-Length:	114
Content-Type:	application/x-www-fitng:
Cookie:	connect.sid=s%age:
Host:	localhost:3001bsl:
Origin:	http://localhve



```
<input
accept="image/*"
<!-- multiple -->
<!-- accept="image/jpg,image/png"  -->
type="file"
name="photo"
class="border-2 rounded-lg p-3 bg-gray-50 focus:ring-2 focus:ring-indigo-400 outline-none"
  />
```



20.3 Handling Multipart Form Data

[redacted]

npm — Search packages — Search — Sign Up — Sign In

multer DT
1.4.5-lts.1 • Public • Published 2 years ago

Readme | Code Beta | 7 Dependencies | 4,749 Dependents | 46 Versions

Multer build unknown · npm package 1.4.5-lts.1 · code style standard

Multer is a node.js middleware for handling `multipart/form-data`, which is primarily used for uploading files. It is written on top of busboy for maximum efficiency.

NOTE: Multer will not process any form which is not multipart (`multipart/form-data`).

Translations

This README is also available in other languages:

- Español (Spanish)
- 简体中文 (Chinese)
- 한국어 (Korean)
- Русский язык (Russian)
- Việt Nam (Vietnam)
- Português (Portuguese Brazil)

Installation

```
$ npm install --save multer
```

Install
```
> npm i multer
```

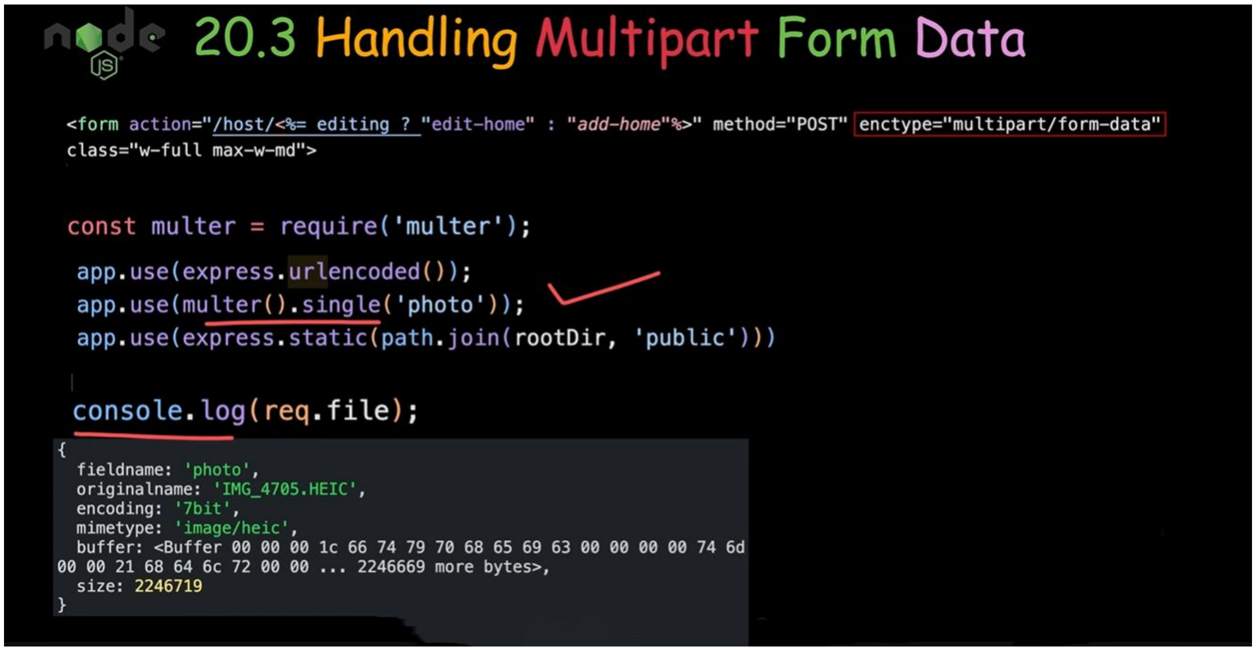
Repository
github.com/expressjs/multer

Homepage
github.com/expressjs/multer#readme

Weekly Downloads
6,313,138

Version	License
1.4.5-lts.1	MIT
Unpacked Size	Total Files
27.6 kB	11
Issues	Pull Requests
195	70

```
prashantjain@Prashants-MacBook-Pro 16 email and advance auth % npm install multer

added 17 packages, and audited 298 packages in 719ms

50 packages are looking for funding
  run `npm fund` for details

1 high severity vulnerability

To address all issues, run:
  npm audit fix

Run `npm audit` for details.
```



20.3 Handling Multipart Form Data

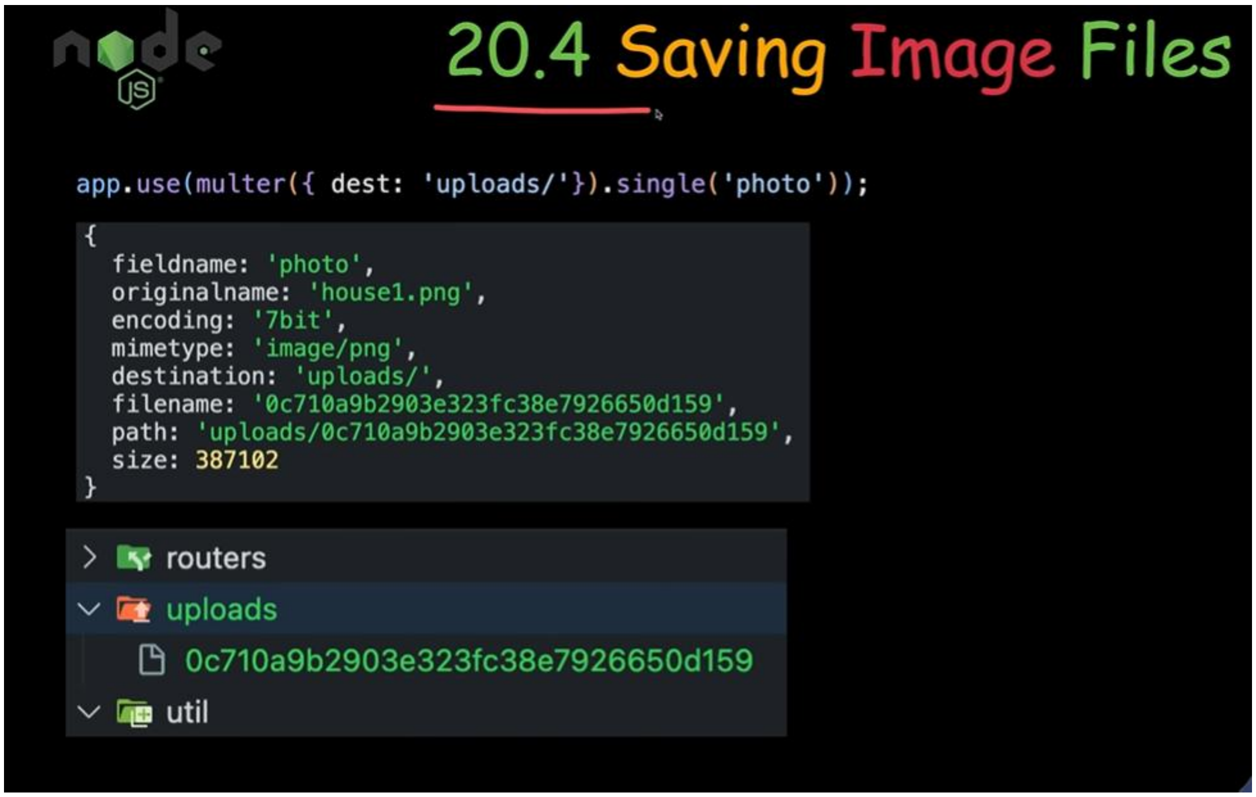
```
<form action="/host/<%= editing ? "edit-home" : "add-home"%>" method="POST" enctype="multipart/form-data"
class="w-full max-w-md">
```

```
const multer = require('multer');
```

```
app.use(express.urlencoded());
app.use(multer().single('photo'));
app.use(express.static(path.join(rootDir, 'public')))
```

```
console.log(req.file);
```

```
{
  fieldname: 'photo',
  originalname: 'IMG_4705.HEIC',
  encoding: '7bit',
  mimetype: 'image/heic',
  buffer: <Buffer 00 00 00 1c 66 74 79 70 68 65 69 63 00 00 00 00 74 6d
00 00 21 68 64 6c 72 00 00 ... 2246669 more bytes>,
  size: 2246719
}
```

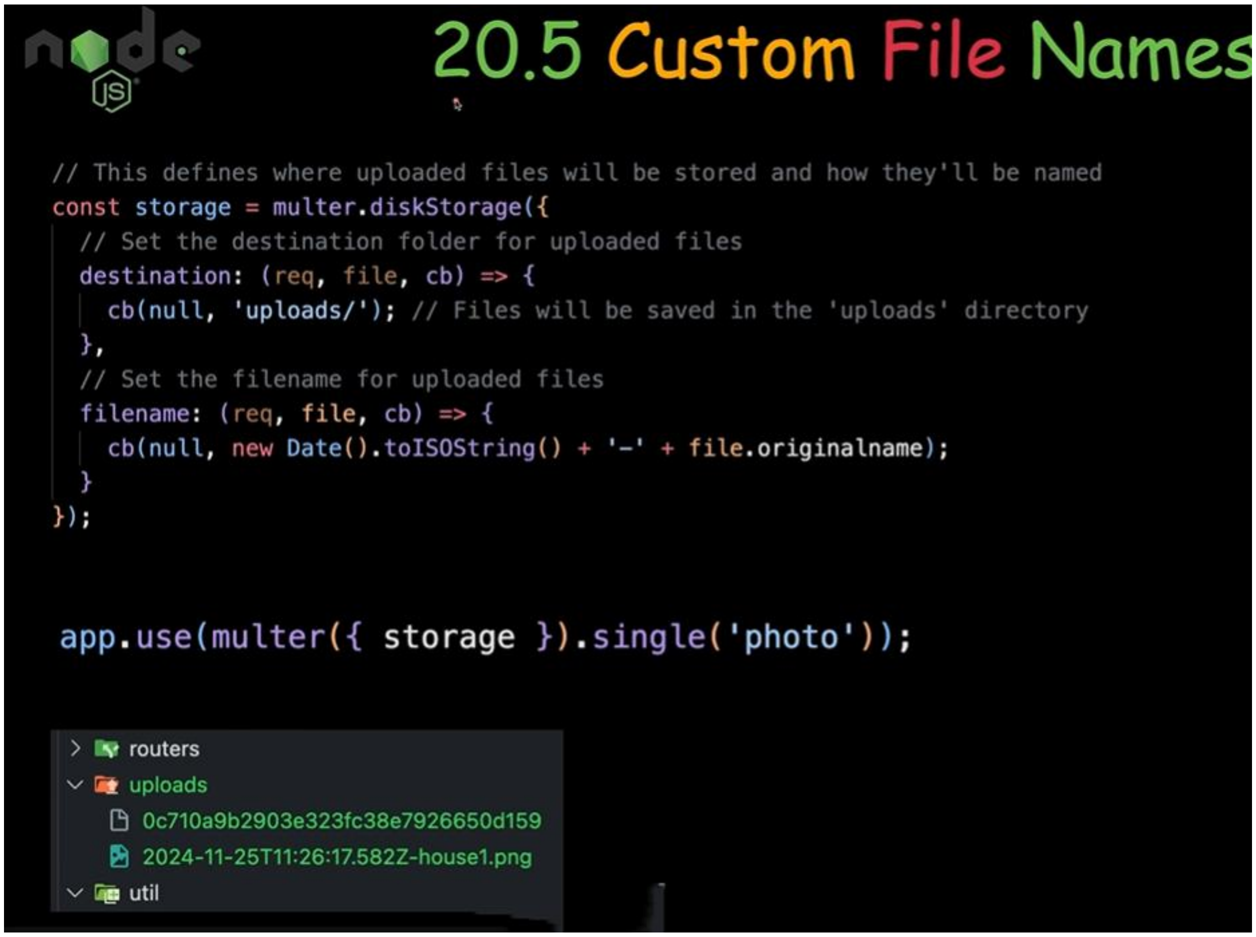


20.4 Saving Image Files

```
app.use(multer({ dest: 'uploads/'}).single('photo'));
```

```
{
  fieldname: 'photo',
  originalname: 'house1.png',
  encoding: '7bit',
  mimetype: 'image/png',
  destination: 'uploads/',
  filename: '0c710a9b2903e323fc38e7926650d159',
  path: 'uploads/0c710a9b2903e323fc38e7926650d159',
  size: 387102
}
```

```
> routers
v uploads
    0c710a9b2903e323fc38e7926650d159
v util
```

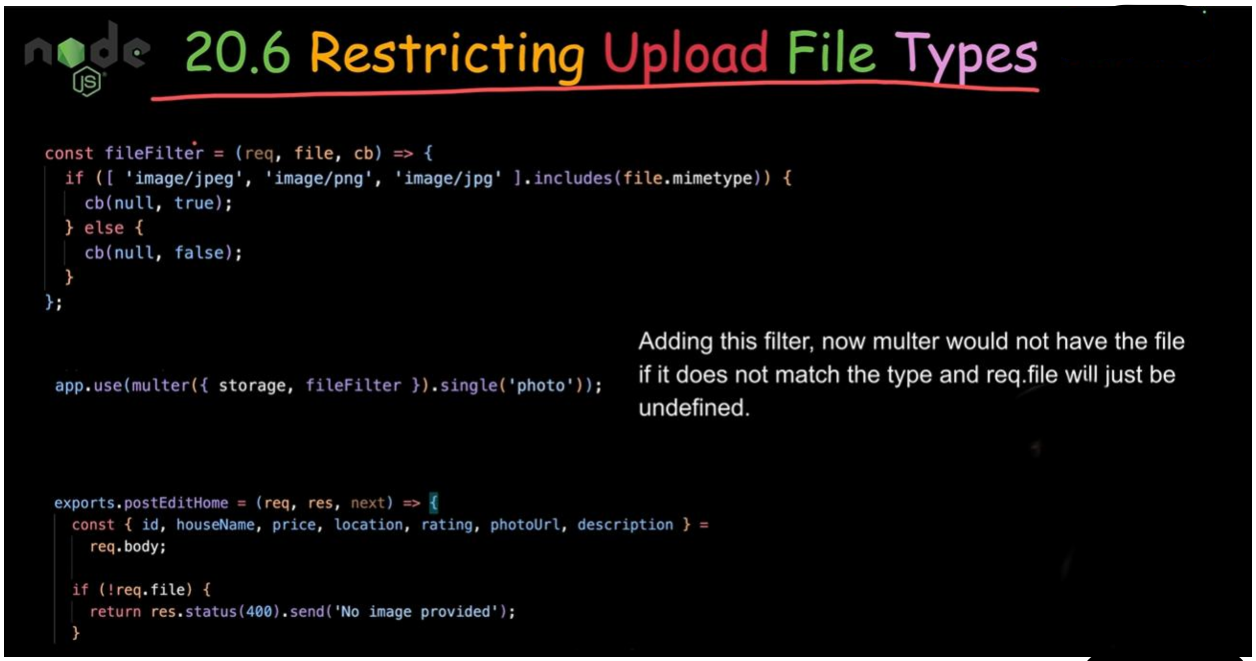


20.5 Custom File Names

```
// This defines where uploaded files will be stored and how they'll be named
const storage = multer.diskStorage({
  // Set the destination folder for uploaded files
  destination: (req, file, cb) => {
    cb(null, 'uploads/'); // Files will be saved in the 'uploads' directory
  },
  // Set the filename for uploaded files
  filename: (req, file, cb) => {
    cb(null, new Date().toISOString() + '-' + file.originalname);
  }
});
```

```
app.use(multer({ storage }).single('photo'));
```

```
> routers
v uploads
    0c710a9b2903e323fc38e7926650d159
    2024-11-25T11:26:17.582Z-house1.png
v util
```



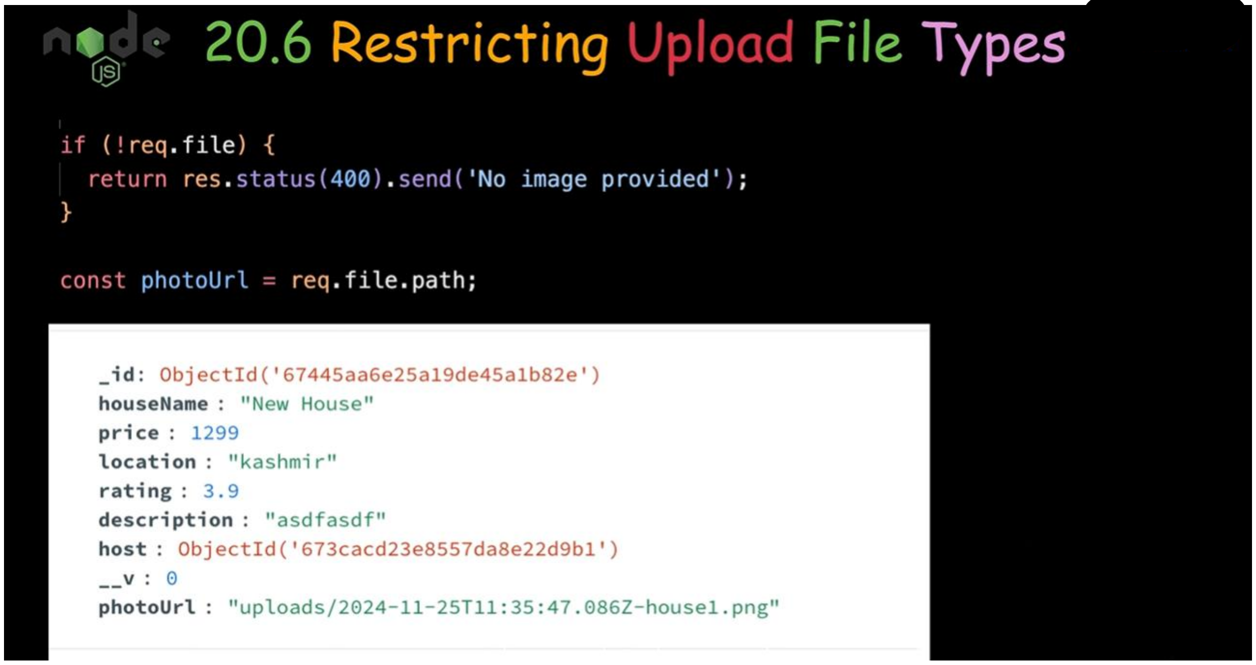
20.6 Restricting Upload File Types

```
const fileFilter = (req, file, cb) => {
  if ([ 'image/jpeg', 'image/png', 'image/jpg' ].includes(file.mimetype)) {
    cb(null, true);
  } else {
    cb(null, false);
  }
};
```

```
app.use(multer({ storage, fileFilter }).single('photo'));
```

Adding this filter, now multer would not have the file if it does not match the type and req.file will just be undefined.

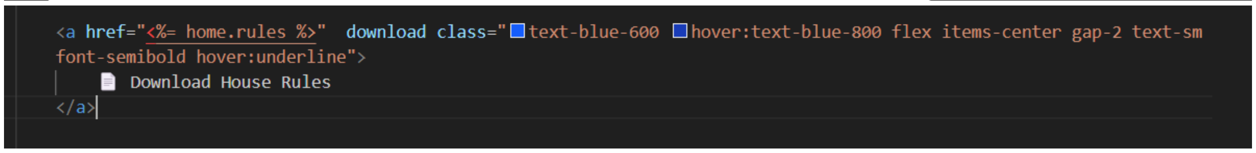
```
exports.postEditHome = (req, res, next) => {
  const { id, houseName, price, location, rating, photoUrl, description } =
    req.body;

  if (!req.file) {
    return res.status(400).send('No image provided');
  }
```



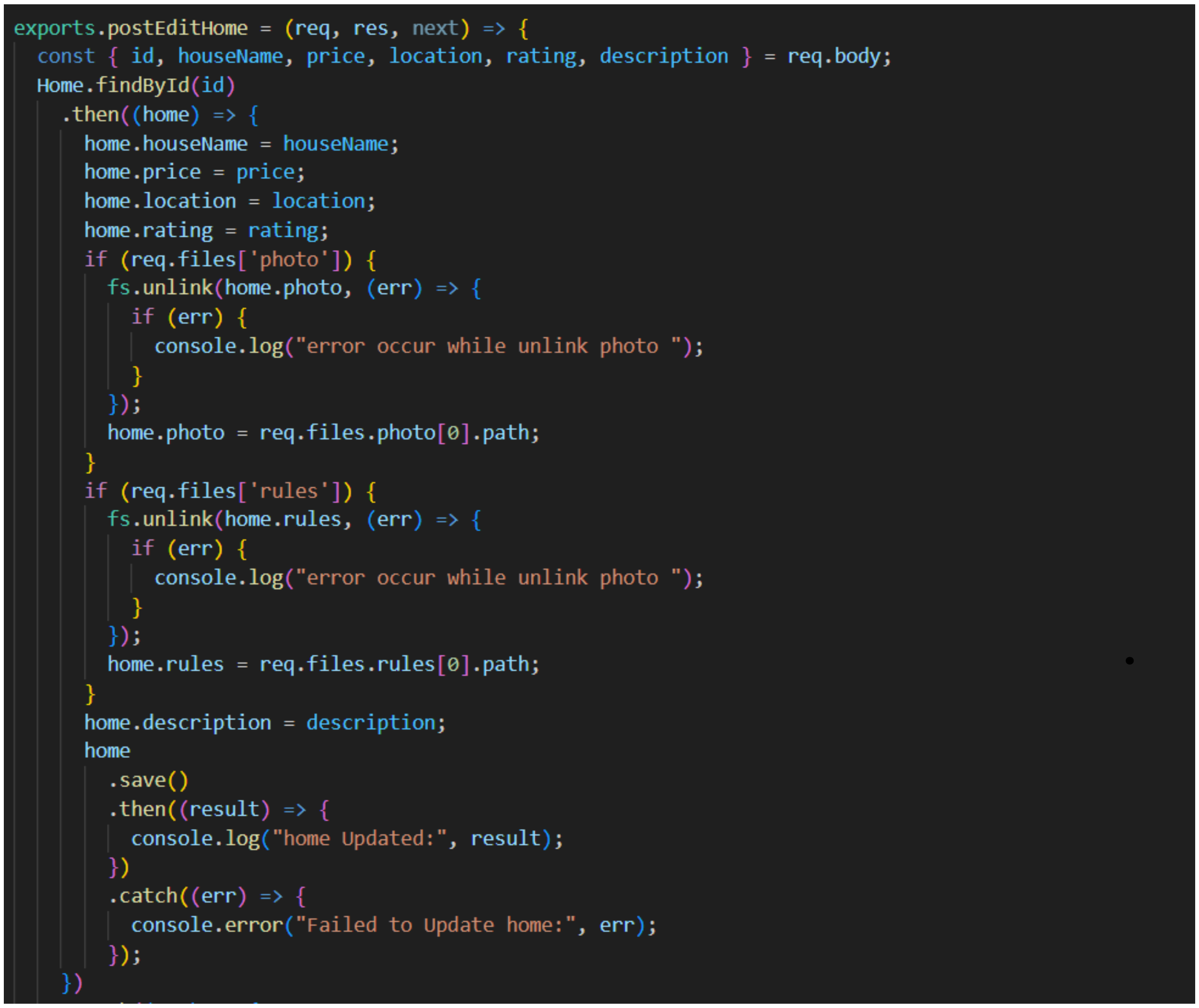
20.6 Restricting Upload File Types

```
if (!req.file) {
  return res.status(400).send('No image provided');
}

const photoUrl = req.file.path;
```

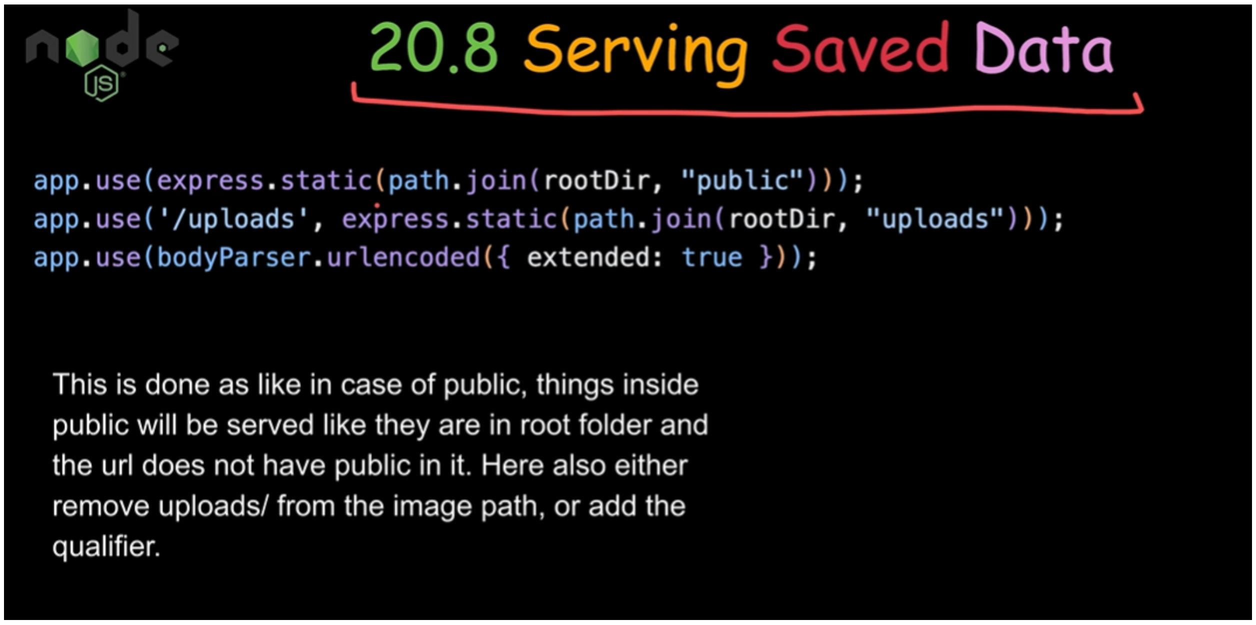
```
_id: ObjectId('67445aa6e25a19de45a1b82e')
houseName : "New House"
price : 1299
location : "kashmir"
rating : 3.9
description : "asdfasdf"
host : ObjectId('673cacd23e8557da8e22d9b1')
__v : 0
photoUrl : "uploads/2024-11-25T11:35:47.086Z-house1.png"
```



```
<a href="<%= home.rules %>"  download class="text-blue-600 hover:text-blue-800 flex items-center gap-2 text-sm
font-semibold hover:underline">
    📄 Download House Rules
</a>
```



```javascript
exports.postEditHome = (req, res, next) => {
  const { id, houseName, price, location, rating, description } = req.body;
  Home.findById(id)
    .then((home) => {
      home.houseName = houseName;
      home.price = price;
      home.location = location;
      home.rating = rating;
      if (req.files['photo']) {
        fs.unlink(home.photo, (err) => {
          if (err) {
            console.log("error occur while unlink photo ");
          }
        });
        home.photo = req.files.photo[0].path;
      }
      if (req.files['rules']) {
        fs.unlink(home.rules, (err) => {
          if (err) {
            console.log("error occur while unlink photo ");
          }
        });
        home.rules = req.files.rules[0].path;
      }
      home.description = description;
      home
        .save()
        .then((result) => {
          console.log("home Updated:", result);
        })
        .catch((err) => {
          console.error("Failed to Update home:", err);
        });
    })
```



20.8 Serving Saved Data

```javascript
app.use(express.static(path.join(rootDir, "public")));
app.use('/uploads', express.static(path.join(rootDir, "uploads")));
app.use(bodyParser.urlencoded({ extended: true }));
```

This is done as like in case of public, things inside public will be served like they are in root folder and the url does not have public in it. Here also either remove uploads/ from the image path, or add the qualifier.