



Upload & resize multiple images in Node.js using Express, Multer, Sharp

📅 Last modified: June 28, 2021 (<https://www.bezkoder.com/node-js-upload-resize-multiple-images/>) 
bezcoder (<https://www.bezkoder.com/author/bezkoder/>)  Node.js
(<https://www.bezkoder.com/category/node-js/>)

In this tutorial, I will show you how to upload & resize multiple images in Node.js using Express, Multer and Sharp.

Related Posts:

- How to upload multiple files in Node.js (<https://bezcoder.com/node-js-upload-multiple-files/>)
- Upload/store images in MySQL using Node.js, Express & Multer (<https://bezcoder.com/node-js-upload-image-mysql/>)
- Node.js Express File Upload Rest API example using Multer (<https://bezcoder.com/node-js-express-file-upload/>)
- Google Cloud Storage with Node.js: File Upload example (<https://bezcoder.com/google-cloud-storage-nodejs-upload-file/>)

Fullstack:

- File Upload using Angular 8 and Node.js example (<https://bezcoder.com/angular-8-node-js-file-upload/>)
- File Upload using Angular 11 and Node.js example (<https://bezcoder.com/angular-11-node-js-file-upload/>)
- File Upload using Angular 12 and Node.js example (<https://bezcoder.com/angular-12-node-js-file-upload/>)

Contents [hide]

Uploading multiple images using Multer

Upload multiple images View

Using Multer to upload multiple Images

Resizing images using Sharp

The Route Handler Functions

Demo

Practice: Upload & Resize multiple images with Node.js

Project Structure

Setup Node.js modules

Create View for uploading multiple images

Create Controller for the view

Create Controller for processing Images

Define routes

Create Express app server

Run the app

Conclusion

Further Reading

There is a couple of things to learn before we go to the practice part.

Uploading multiple images using Multer

Upload multiple images View

In HTML, we create a form with following elements:

- `action="/multiple-upload"`
- `enctype="multipart/form-data"`
- `multiple` input

```
<form
  action="/multiple-upload"
  method="POST"
  enctype="multipart/form-data">
  ...
  <input type="file" multiple>
  ...
</form>
```

Using Multer to upload multiple Images

Let's configure **multer** and create some functions.

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

const multerStorage = multer.memoryStorage();

const multerFilter = (req, file, cb) => {
  if (file.mimetype.startsWith("image")) {
    cb(null, true);
  } else {
    cb("Please upload only images.", false);
  }
};

const upload = multer({
  storage: multerStorage,
  fileFilter: multerFilter
});

const uploadFiles = upload.array("images", 10); // limit to 10 images

const uploadImages = (req, res, next) => {
  uploadFiles(req, res, err => {
    if (err instanceof multer.MulterError) { // A Multer error occurred when uploading.
      if (err.code === "LIMIT_UNEXPECTED_FILE") { // Too many images exceeding the allowe
        // ...
      }
    } else if (err) {
      // handle other errors
    }

    // Everything is ok.
    next();
  });
};
```

In the code above, we've done these steps:

- First, import `multer` module.
- Next, configure `multer` to store images in memory as a buffer so that we could then use it later by other processes (resize the images before saving them to disk).
- We also define a filter to only allow images to pass.
- Then use `multer` object to create `uploadFiles` function with the field `images` and the max count. The array of files will be stored in `req.files`.
- We catch errors by calling the `uploadFiles` middleware function, if there is no error, call `next()`.

Resizing images using Sharp

Now we're gonna process all of uploaded images (which are in memory).

```

const sharp = require("sharp");

const resizeImages = async (req, res, next) => {
  if (!req.files) return next();

  req.body.images = [];
  await Promise.all(
    req.files.map(async file => {
      const newFilename = ...;

      await sharp(file.buffer)
        .resize(640, 320)
        .toFormat("jpeg")
        .jpeg({ quality: 90 })
        .toFile(`upload/${newFilename}`);

      req.body.images.push(newFilename);
    })
  );

  next();
};

```

Let me explain the code.

– We import `sharp` module.

– The `resizeImages` will be an async function in that:

- We check if there are no images uploaded, move straight to the next middleware
- For each image, we return a `Promise`, we need all of the `Promises` done before calling `next()`. So we get an array of all of these `Promises` using `array.map()` then put it in `Promise.all()` function.
- We process every image in the `images` array using `resize()` (<http://sharp.pixelplumbing.com/en/stable/api-resize/>) method, then change the format (with file extension) to `jpeg` with `toFormat()` (<http://sharp.pixelplumbing.com/en/stable/api-output/#toformat>) and save it to disk using `toFile()` (<http://sharp.pixelplumbing.com/en/stable/api-output/#tofile>) method.

The Route Handler Functions

We write all middleware functions above in a Controller called `upload`.

We use Express `Router` to handle POST action to `"/multiple-upload"` endpoint.

```
const express = require("express");
const router = express.Router();
const uploadController = require("../controllers/upload");

let routes = app => {
  ...
  router.post(
    "/multiple-upload",
    uploadController.uploadImages,
    uploadController.resizeImages,
    uploadController.getResult
  );

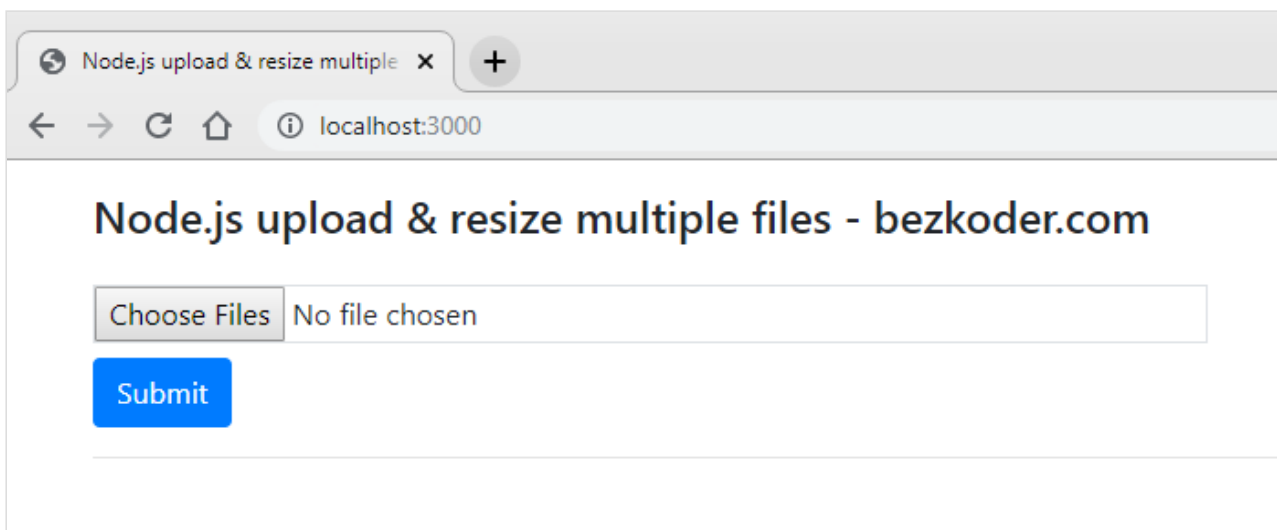
  return app.use("/", router);
};

module.exports = routes;
```

You can see that we insert `uploadImages` and `resizeImages`, then `uploadController.getResult`. This is because we will process images before showing the result message.

Demo

Open browser for the app:



Choose some images to upload & resize:

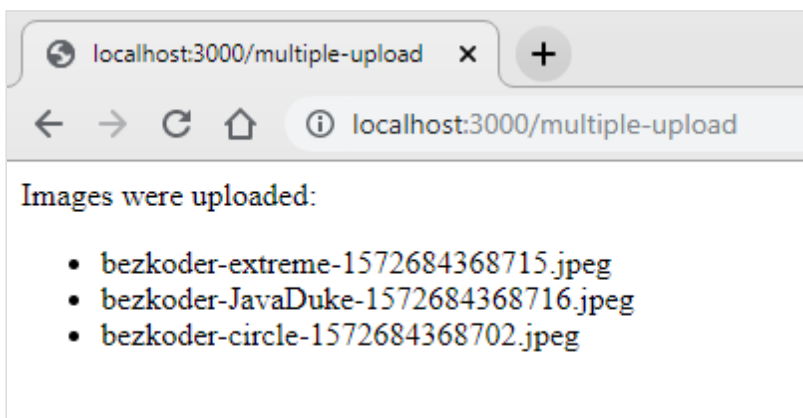
Node.js upload & resize multiple files - bezkoder.com

Choose Files 3 files

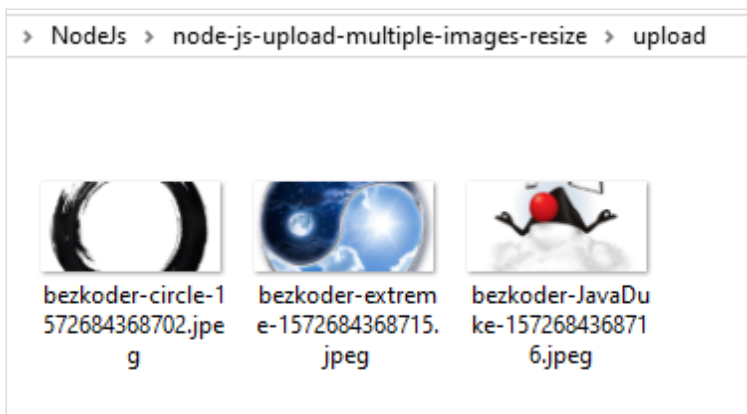
Submit



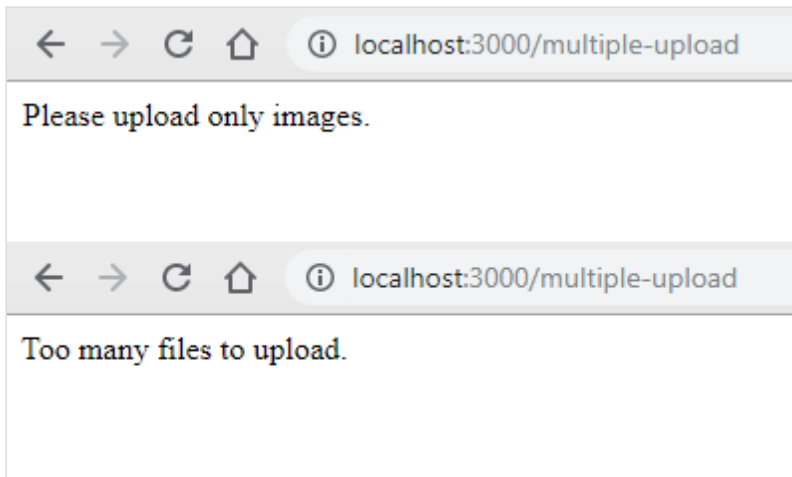
Click on **Submit** button, if the process is successful, browser shows the uploaded images:



Check the **upload** folder, you can see the resized images:



If you choose the file without image format, or the images you want to upload exceeds the allowed amount, the browser will show the messages.

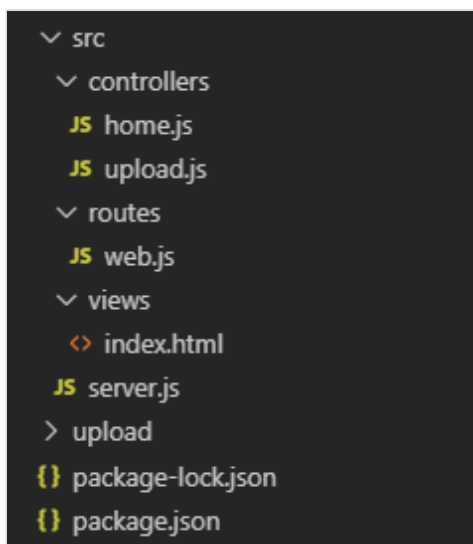


Practice: Upload & Resize multiple images with Node.js

Now I'm gonna show you how to build this Node.js application step by step.

Project Structure

This is our project structure:



- upload : the folder for storing uploaded images.
- views/index.html : contains HTML form for user to upload images.
- routes/web.js : defines routes for endpoints that is called from views, use controllers to handle requests.
- controllers :
 - home.js returns views/index.html
 - upload.js handles upload & resize multiple images with middleware functions.
- server.js : initializes routes, runs Express app.

Setup Node.js modules

Change current directory to the root folder of our project, then install Express, Multer, Sharp with the following command:

```
npm install express multer sharp
```

Create View for uploading multiple images

In **views** folder, create `index.html` with the HTML and Javascript code as below:


```

<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta http-equiv="X-UA-Compatible" content="ie=edge" />
    <title>Node.js upload & resize multiple files</title>
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/b
    <style>
      div.preview-images > img {
        width: 30%;
      }
    </style>
  </head>
  <body>
    <div class="container">
      <div class="row">
        <div class="col-sm-8 mt-3">
          <h4>Node.js upload & resize multiple files - bezkoder.com</h4>

          <form class="mt-4"
            action="/multiple-upload"
            method="POST"
            enctype="multipart/form-data"
          >
            <input
              type="file"
              name="images"
              multiple
              id="input-images"
              class="form-control-file border"
            />
            <button type="submit" class="btn btn-primary mt-2">Submit</button>
          </form>
        </div>
      </div>
      <hr />
      <div class="row">
        <div class="col-sm-12">
          <div class="preview-images"></div>
        </div>
      </div>
    </div>
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.bundle.m
    <script src="https://code.jquery.com/jquery-3.4.1.min.js"></script>
    <script>
      $(document).ready(function() {
        let imagesPreview = function(input, placeToInsertImagePreview) {
          if (input.files) {

```

```

    let filesAmount = input.files.length;
    for (i = 0; i < filesAmount; i++) {
        let reader = new FileReader();
        reader.onload = function(event) {
            $($.parseHTML("<img>"))
                .attr("src", event.target.result)
                .appendTo(placeToInsertImagePreview);
        };
        reader.readAsDataURL(input.files[i]);
    }
}
});
$("#input-images").on("change", function() {
    imagesPreview(this, "div.preview-images");
});
});
</script>
</body>
</html>

```

The HTML code is easy to understand, the jQuery script shows preview of the chosen images. We also use Bootstrap to make the UI more comfortable to read.

Create Controller for the view

Under **controllers** folder, create `home.js` file.

```

const path = require("path");

const home = (req, res) => {
    return res.sendFile(path.join(`${__dirname}/../views/index.html`));
};

module.exports = {
    getHome: home
};

```

Create Controller for processing Images

This is the main code of our Node.js project. The Upload Controller uses both **multer** & **sharp** module.

We're gonna export 3 middleware functions:

- `uploadImages`
- `resizeImages`
- `getResult`

```
const multer = require("multer");
const sharp = require("sharp");

const multerStorage = multer.memoryStorage();

const multerFilter = (req, file, cb) => {
  if (file.mimetype.startsWith("image")) {
    cb(null, true);
  } else {
    cb("Please upload only images.", false);
  }
};

const upload = multer({
  storage: multerStorage,
  fileFilter: multerFilter
});

const uploadFiles = upload.array("images", 10);

const uploadImages = (req, res, next) => {
  uploadFiles(req, res, err => {
    if (err instanceof multer.MulterError) {
      if (err.code === "LIMIT_UNEXPECTED_FILE") {
        return res.send("Too many files to upload.");
      }
    } else if (err) {
      return res.send(err);
    }

    next();
  });
};

const resizeImages = async (req, res, next) => {
  if (!req.files) return next();

  req.body.images = [];
  await Promise.all(
    req.files.map(async file => {
      const filename = file.originalname.replace(/\.\.+$/, "");
      const newFilename = `bezcoder-${filename}-${Date.now()}.jpeg`;

      await sharp(file.buffer)
        .resize(640, 320)
        .toFormat("jpeg")
        .jpeg({ quality: 90 })
        .toFile(`upload/${newFilename}`);
    })
  );
}
```

```
    req.body.images.push(newFilename);
  })
);

next();
};

const getResult = async (req, res) => {
  if (req.body.images.length <= 0) {
    return res.send(`You must select at least 1 image.`);
  }

  const images = req.body.images
    .map(image => "" + image + "")
    .join("");

  return res.send(`Images were uploaded:${images}`);
};

module.exports = {
  uploadImages: uploadImages,
  resizeImages: resizeImages,
  getResult: getResult
};
```

In the `resizeImages` function, we:

- get each image name and create new name for it with `jpeg` extension.
- resize to `640x320`
- change the image format to `jpeg`
- set the quality to 90% of the original image
- save the image file to **upload** folder

Define routes

We define routes in `routes/web.js` like this.

There are 2 routes:

- GET: Home page for the upload form.
- POST `"/multiple-upload"` to use three `uploadController` functions.

```
const express = require("express");
const router = express.Router();
const homeController = require("../controllers/home");
const uploadController = require("../controllers/upload");

let routes = app => {
  router.get("/", homeController.getHome);

  router.post(
    "/multiple-upload",
    uploadController.uploadImages,
    uploadController.resizeImages,
    uploadController.getResult
  );

  return app.use("/", router);
};

module.exports = routes;
```

Create Express app server

Finally, we only need to create an Express server.

server.js

```
const express = require("express");
const app = express();
const initRoutes = require("./routes/web");

app.use(express.urlencoded({ extended: true }));
initRoutes(app);

let port = 3000;
app.listen(port, () => {
  console.log(`Running at localhost:${port}`);
});
```

Run the app

Before running this Node.js app, you must create **upload** folder in the project root folder.

Run following command:

```
node src/server.js
```

If everything is ok, the console shows:

```
Running at localhost:3000
```

Open browser with url <http://localhost:3000/> and you can check the result now.

Conclusion

Today we've learned way to upload multiple images using Express and Multer, we also know how to limit the number of images, resize these images and change the extensions using Sharp.

Happy learning! See you again.

Further Reading

- <https://www.npmjs.com/package/multer> (<https://www.npmjs.com/package/multer>)
- <https://www.npmjs.com/package/sharp> (<https://www.npmjs.com/package/sharp>)
- <https://www.npmjs.com/package/express> (<https://www.npmjs.com/package/express>)

Security:

- Node.js & MySQL: User Authentication & Authorization (<https://bezcoder.com/node-js-jwt-authentication-mysql/>)
- Node.js & PostgreSQL: User Authentication & Authorization (<https://bezcoder.com/node-js-jwt-authentication-postgresql/>)
- Node.js + MongoDB: User Authentication & Authorization (<https://bezcoder.com/node-js-mongodb-auth-jwt/>)

[express](https://www.bezkoder.com/tag/express/) (<https://www.bezkoder.com/tag/express/>) [file](https://www.bezkoder.com/tag/file/) (<https://www.bezkoder.com/tag/file/>)

[multer](https://www.bezkoder.com/tag/multer/) (<https://www.bezkoder.com/tag/multer/>) [node.js](https://www.bezkoder.com/tag/node-js/) (<https://www.bezkoder.com/tag/node-js/>)

[sharp](https://www.bezkoder.com/tag/sharp/) (<https://www.bezkoder.com/tag/sharp/>) [upload](https://www.bezkoder.com/tag/upload/) (<https://www.bezkoder.com/tag/upload/>)

23 thoughts to “Upload & resize multiple images in Node.js using Express, Multer, Sharp”



oshea

November 21, 2019 at 10:45 am (<https://www.bezkoder.com/node-js-upload-resize-multiple-images/#comment-48>)

Hello, I enjoy reading this Node.js tutorial, it's clear and easy to understand. Thank you!

REPLY



teropro

November 21, 2019 at 5:12 pm (<https://www.bezkoder.com/node-js-upload-resize-multiple-images/#comment-49>)

Perfectly composed post! Thanks.

REPLY



trishaba

November 21, 2019 at 11:52 pm (<https://www.bezkoder.com/node-js-upload-resize-multiple-images/#comment-52>)

Good explanation for working with images in Node.js. Thanks.

REPLY



hanks

November 23, 2019 at 9:18 pm (<https://www.bezkoder.com/node-js-upload-resize-multiple-images/#comment-61>)

This is what I need, I run the Node App and it works. Thanks!

REPLY



Pradeesh

February 6, 2020 at 12:59 pm (<https://www.bezkoder.com/node-js-upload-resize-multiple-images/#comment-310>)

Hey there thanks for the awesome tutorial

I tried out the above code the size of the image is reduced but while I upload an image with transparent background the outcome of the images isn't transparent the background changes to white.

Is there any way I can achieve the desired outcome.

Thanks in advance...!!

REPLY



bezkoder

February 7, 2020 at 3:14 pm (<https://www.bezkoder.com/node-js-upload-resize-multiple-images/#comment-326>)

Hi, we cannot make a JPEG image transparent. So you can use `sharp` to resize image to a format that allows transparency, like GIF or PNG.

REPLY

**Berry Litter**

April 21, 2020 at 8:02 pm (<https://www.bezkoder.com/node-js-upload-resize-multiple-images/#comment-1068>)

I enjoy this Node.js tutorial a lot, thanks!

[REPLY](#)**MJ**

May 8, 2020 at 8:52 am (<https://www.bezkoder.com/node-js-upload-resize-multiple-images/#comment-2098>)

Excellent tutorial!

I can find this project on your GitHub page. Can you make it available to download?

[REPLY](#)**daebert**

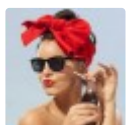
May 14, 2020 at 12:40 am (<https://www.bezkoder.com/node-js-upload-resize-multiple-images/#comment-2413>)

Many thanks! Your tutorial helps me alot.

[REPLY](#)**tito**

May 17, 2020 at 12:28 pm (<https://www.bezkoder.com/node-js-upload-resize-multiple-images/#comment-2518>)

just to tell you I LOVE YOU. thanks

[REPLY](#)**tito**

May 17, 2020 at 12:29 pm (<https://www.bezkoder.com/node-js-upload-resize-multiple-images/#comment-2519>)

it will be amazing if you know how to rotate images from mobile with this modules

[REPLY](#)

**shan**

May 19, 2020 at 1:50 pm (<https://www.bezkoder.com/node-js-upload-resize-multiple-images/#comment-2636>)

Thanks! This Nodejs tutorial helps me so much.

[REPLY](#)**Peter**

August 20, 2020 at 12:31 am (<https://www.bezkoder.com/node-js-upload-resize-multiple-images/#comment-4388>)

I always enjoy your tutorial.

Thanks for guiding me.

[REPLY](#)**Nishchay Gupta**

September 16, 2020 at 6:06 pm (<https://www.bezkoder.com/node-js-upload-resize-multiple-images/#comment-4757>)

Thanks for this lovely tutorial. It is quite helpful. It would be great if you could create a tutorial for uploading, resizing and storing image in my-sql.

[REPLY](#)**Vett**

December 14, 2020 at 6:22 pm (<https://www.bezkoder.com/node-js-upload-resize-multiple-images/#comment-5779>)

One of the best tutorials I have ever seen !

[REPLY](#)**Whitea**

March 22, 2021 at 3:58 am (<https://www.bezkoder.com/node-js-upload-resize-multiple-images/#comment-7708>)

One of the best Node.js tutorial I have ever seen !

[REPLY](#)

olska

March 28, 2021 at 10:29 pm (<https://www.bezkoder.com/node-js-upload-resize-multiple-images/#comment-7787>)

As a Newbie, I am always browsing online for articles that can benefit me. Thank you

REPLY

Beasley

April 7, 2021 at 11:05 am (<https://www.bezkoder.com/node-js-upload-resize-multiple-images/#comment-7957>)

One of the best tutorials I have ever seen !

REPLY

andres

April 12, 2021 at 12:57 am (<https://www.bezkoder.com/node-js-upload-resize-multiple-images/#comment-8046>)

Thanks!

REPLY

Barbera Renton

May 13, 2021 at 7:41 am (<https://www.bezkoder.com/node-js-upload-resize-multiple-images/#comment-8505>)

Thanks!

REPLY

shiatsu

May 27, 2021 at 1:03 am (<https://www.bezkoder.com/node-js-upload-resize-multiple-images/#comment-8727>)

Best File Upload with Node.js tutorial I have ever seen! Thanks!

REPLY

Bischel

June 1, 2021 at 8:01 am (<https://www.bezkoder.com/node-js-upload-resize-multiple-images/#comment-8807>)

One of the best Nodejs tutorials I have ever seen!

[REPLY](#)**Mensc**

August 17, 2021 at 1:14 pm (<https://www.bezkoder.com/node-js-upload-resize-multiple-images/#comment-10943>)

To be honest, your tutorials are really nice, keep it up! I'll go ahead and bookmark your website to come back in the future. Many thanks!

[REPLY](#)**Leave a Reply**

Your email address will not be published. Required fields are marked *

Comment

Name *

Email *

Website

☐

Save my name, email, and website in this browser for the next time I comment.

POST COMMENT

◀ Integrate React with Node.js Express on same Server/Port (<https://www.bezkoder.com/integrate-react-express-same-server-port/>)

Vuetify Multiple Images Upload example ▶ (<https://www.bezkoder.com/vuetify-multiple-image-upload/>)



FOLLOW US

(htt

ps://

ww

w.yo

utub

e.co

m/c

han



nel/



(htt UCp (htt

ps:// 0mx ps://

face 9RH gith

boo 0Jxa ub.c

k.co Fsm om/

m/b MvK bezk

ezko XA8 oder

der) 6Q))

TOOLSJson Formatter (<https://www.bezkoder.com/json-formatter/>)

(<https://www.dmca.com/Protection/Status.aspx?ID=3f543dd5-c6d8-4208-9a6b-0e92057fd597&refurl=https://www.bezkoder.com/node-js-upload-resize-multiple-images/>)

[Home \(https://bezkoder.com/\)](https://bezkoder.com/)

[Privacy Policy \(https://www.bezkoder.com/privacy-policy/\)](https://www.bezkoder.com/privacy-policy/)

[Contact Us \(https://www.bezkoder.com/contact-us/\)](https://www.bezkoder.com/contact-us/)

[About Us \(https://www.bezkoder.com/about/\)](https://www.bezkoder.com/about/)

BezKoder 2019