Express.js

Other Concepts

Vraj Suratwala

Department of ICT, VNSGU, Surat

Topics

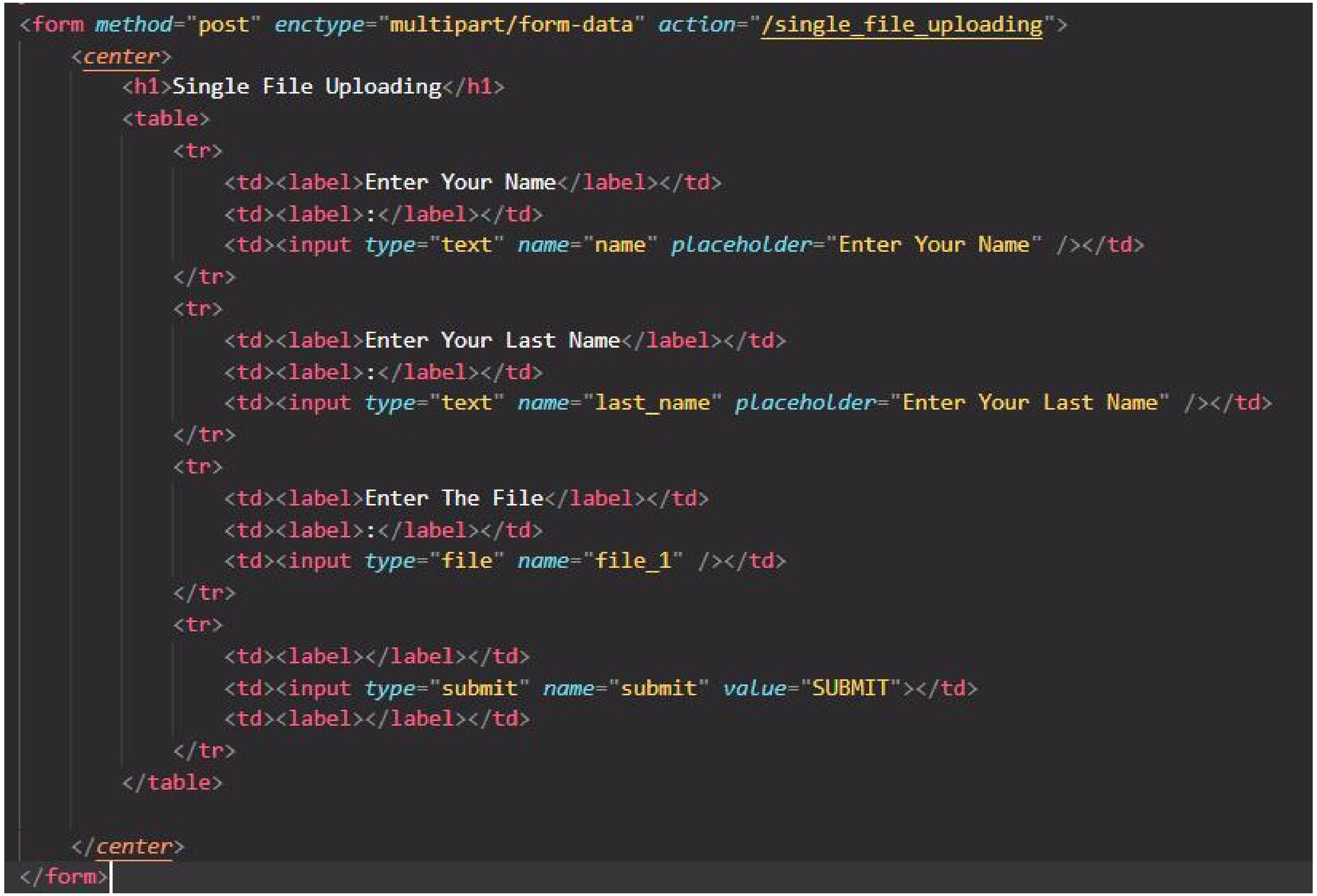
- multer middleware in express.js.
- serving static files in express!
- express validator
- template engine

Multer with Express.js

- 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.
- Installation
  - $ npm install multer
- Usage
  - Multer adds a body object and a file or files object to the request object. The body object contains the values of the text fields of the form, the file or files object contains the files uploaded via the form.

Example : Multer use in Express.js

- index.js

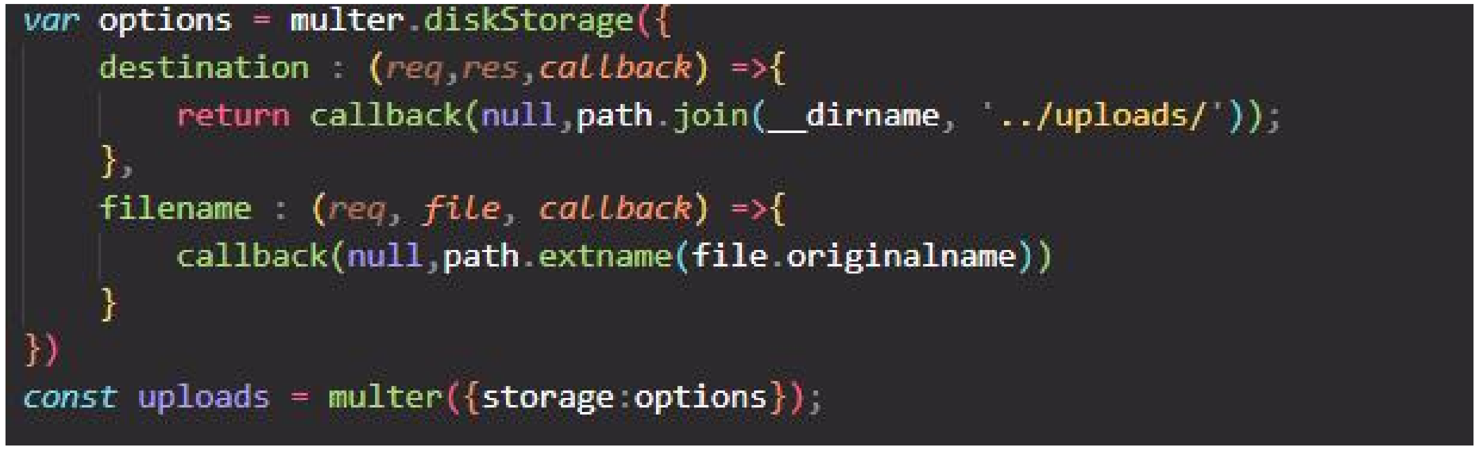


```html
<form method="post" enctype="multipart/form-data" action="/single_file_uploading">
    <center>
        <h1>Single File Uploading</h1>
        <table>
            <tr>
                <td><label>Enter Your Name</label></td>
                <td><label>:</label></td>
                <td><input type="text" name="name" placeholder="Enter Your Name" /></td>
            </tr>
            <tr>
                <td><label>Enter Your Last Name</label></td>
                <td><label>:</label></td>
                <td><input type="text" name="last_name" placeholder="Enter Your Last Name" /></td>
            </tr>
            <tr>
                <td><label>Enter The File</label></td>
                <td><label>:</label></td>
                <td><input type="file" name="file_1" /></td>
            </tr>
            <tr>
                <td><label></label></td>
                <td><input type="submit" name="submit" value="SUBMIT"></td>
                <td><label></label></td>
            </tr>
        </table>

    </center>
</form>
```

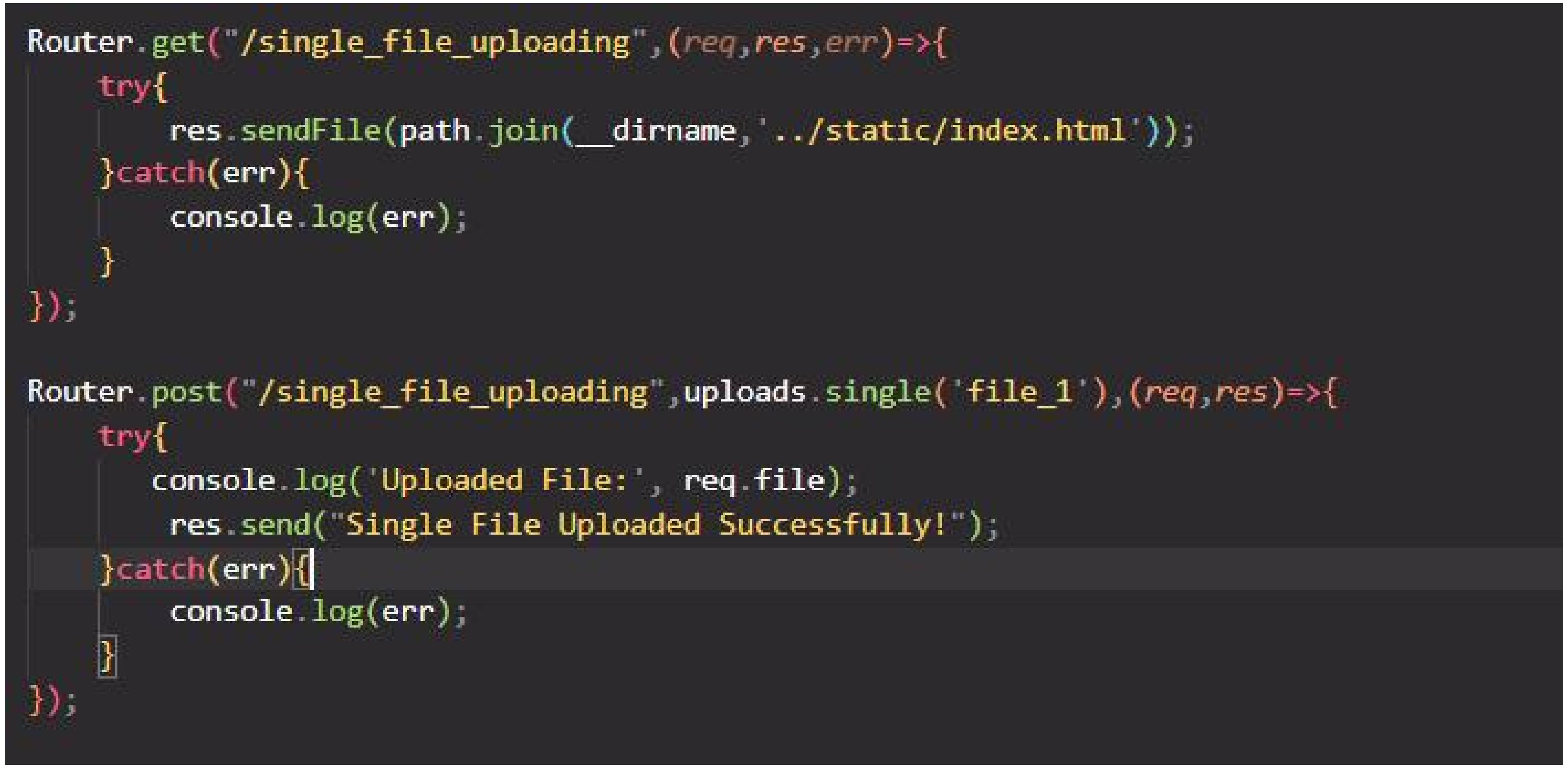
router.js

Basic Configuration as given in the first picture!

then handling the request and uploading the file!

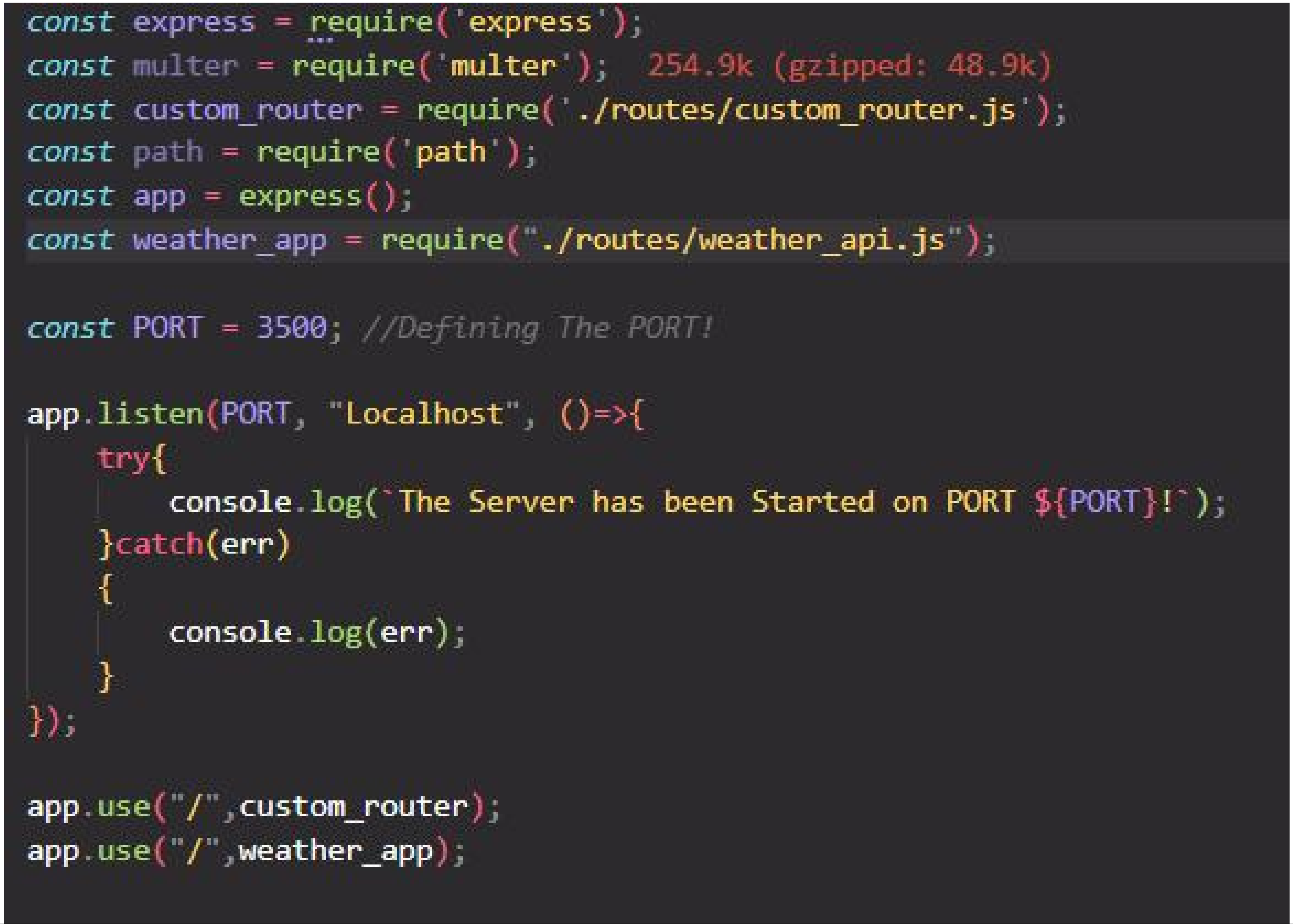


```
var options = multer.diskStorage({
    destination : (req,res,callback) =>{
        return callback(null,path.join(__dirname, '../uploads/'));
    },
    filename : (req, file, callback) =>{
        callback(null,path.extname(file.originalname))
    }
})
const uploads = multer({storage:options});
```



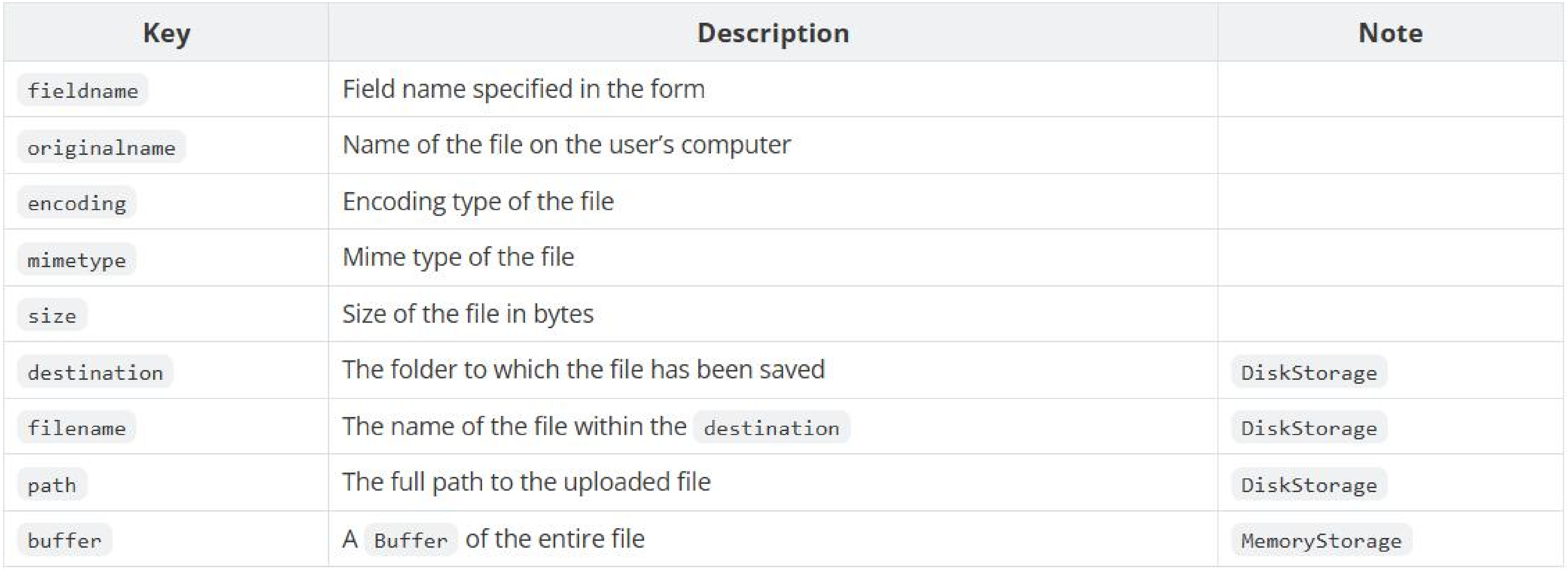
```
Router.get("/single_file_uploading",(req,res,err)=>{
    try{
        res.sendFile(path.join(__dirname,'../static/index.html'));
    }catch(err){
        console.log(err);
    }
});

Router.post("/single_file_uploading",uploads.single('file_1'),(req,res)=>{
    try{
       console.log('Uploaded File:', req.file);
        res.send("Single File Uploaded Successfully!");
    }catch(err){
        console.log(err);
    }
});
```

server.js - main server file which is going to be execute!



```
const express = require('express');
const multer = require('multer');  254.9k (gzipped: 48.9k)
const custom_router = require('./routes/custom_router.js');
const path = require('path');
const app = express();
const weather_app = require("./routes/weather_api.js");

const PORT = 3500; //Defining The PORT!

app.listen(PORT, "Localhost", ()=>{
    try{
        console.log(`The Server has been Started on PORT ${PORT}!`);
    }catch(err)
    {
        console.log(err);
    }
});

app.use("/",custom_router);
app.use("/",weather_app);
```

File Information



Key	Description	Note
`fieldname`	Field name specified in the form	
`originalname`	Name of the file on the user's computer	
`encoding`	Encoding type of the file	
`mimetype`	Mime type of the file	
`size`	Size of the file in bytes	
`destination`	The folder to which the file has been saved	`DiskStorage`
`filename`	The name of the file within the `destination`	`DiskStorage`
`path`	The full path to the uploaded file	`DiskStorage`
`buffer`	A `Buffer` of the entire file	`MemoryStorage`

Reference : https://expressjs.com/en/resources/middleware/multer.html

Serving Static Files in The Express.js

- To serve static files such as images, CSS files, and JavaScript files, use the express.static built-in middleware function in Express.
- for that we have one Middleware as well that is :
  - express.static(root, [options])
  - app.use(express.static('public'))
    - where public is directory name!

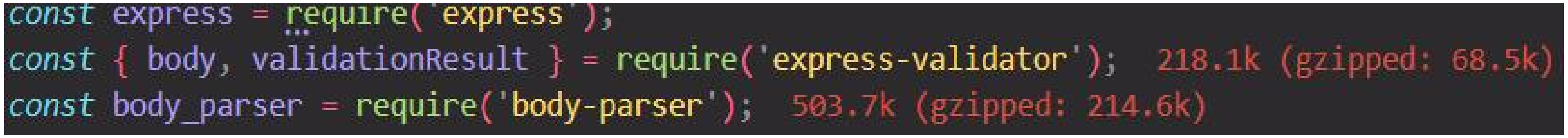
- app.use('/static', express.static('public'))
- Now, you can load the files that are in the public directory from the /static path prefix.
- http://localhost:3000/static/images/kitten.jpg
- http://localhost:3000/static/css/style.css
- http://localhost:3000/static/js/app.js

Express Validator

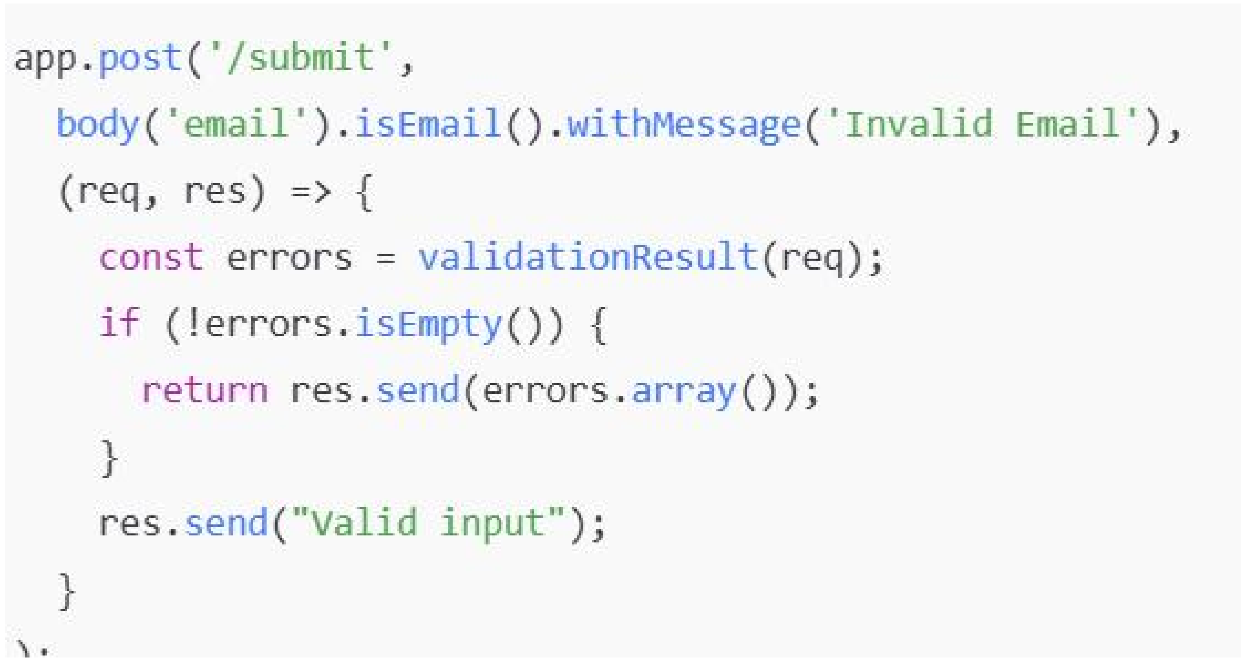
- Express Validator is a middleware for Express.js that helps validate and sanitize user input from forms, APIs, or any HTTP requests.
- It's built on top of the popular validator.js library and integrates smoothly with Express routes.
- Key features :
  - validation of different input type
  - Sanitizes Input
  - Suports Custom Validation Logic
  - Easy to use with route handlers.

Basic Syntax :

For this we have tom import this modules :

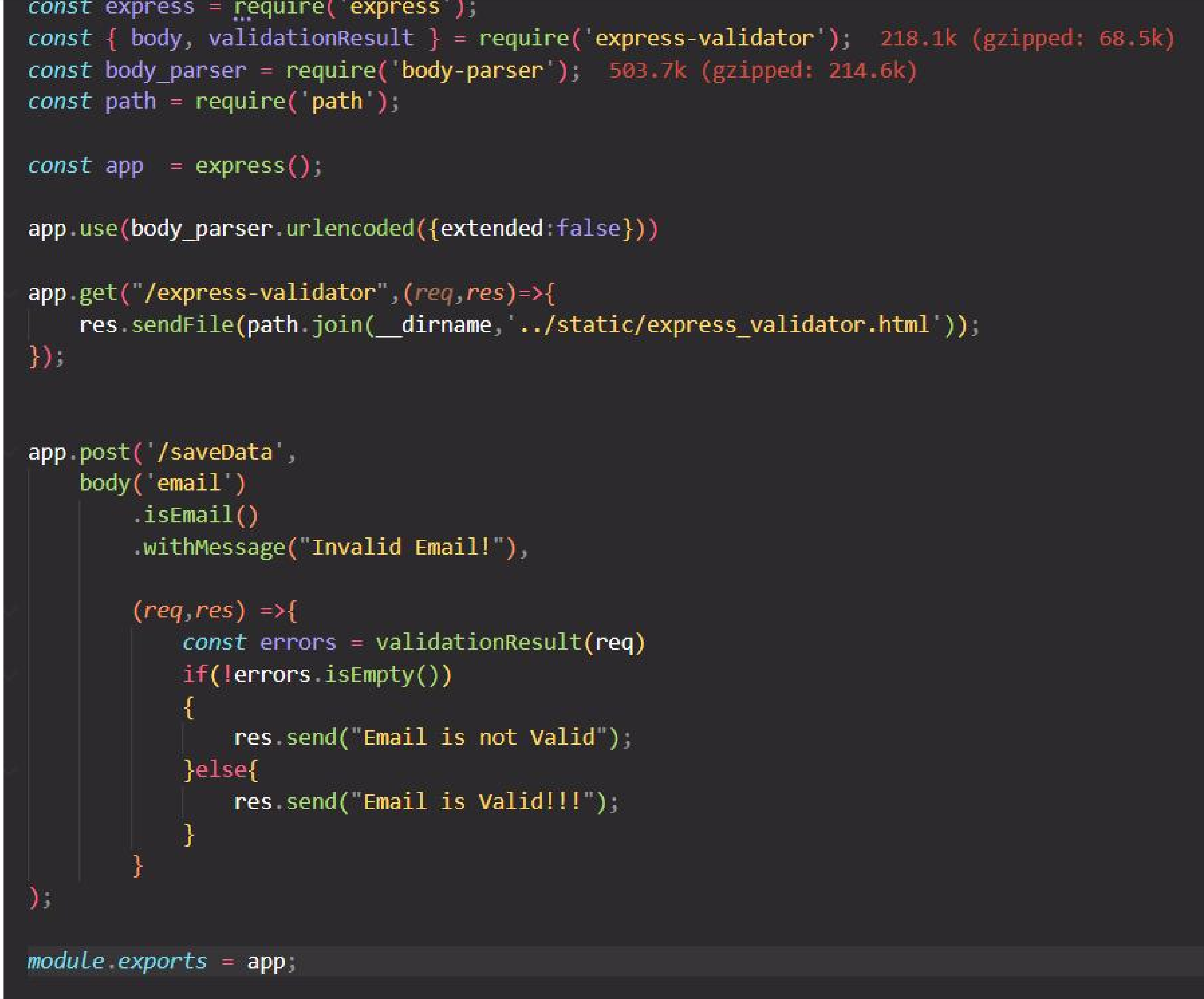


```
const express = require('express');
const { body, validationResult } = require('express-validator');  218.1k (gzipped: 68.5k)
const body_parser = require('body-parser');  503.7k (gzipped: 214.6k)
```



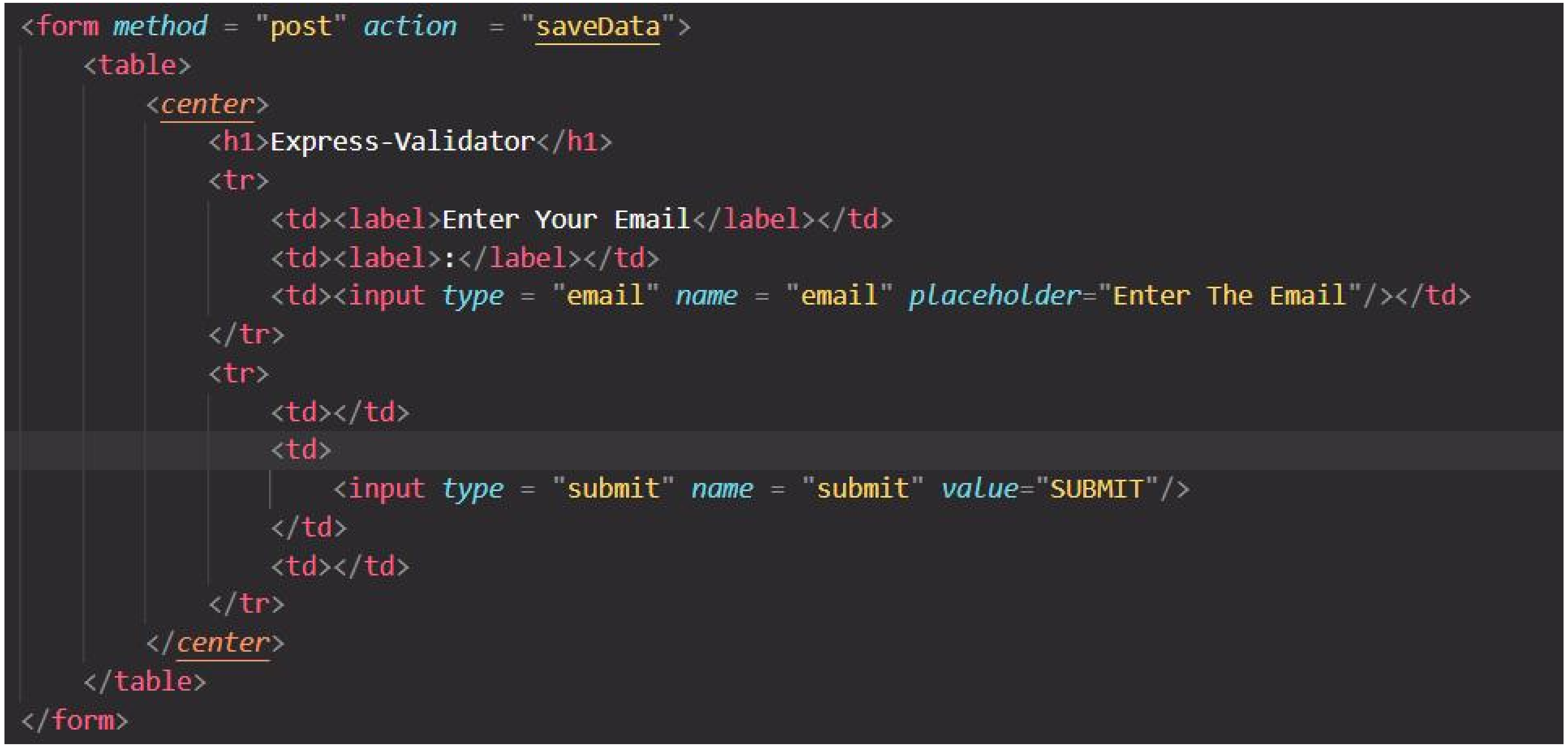
```
app.post('/submit',
  body('email').isEmail().withMessage('Invalid Email'),
  (req, res) => {
    const errors = validationResult(req);
    if (!errors.isEmpty()) {
      return res.send(errors.array());
    }
    res.send("Valid input");
  }
```

Example : express-validator.js



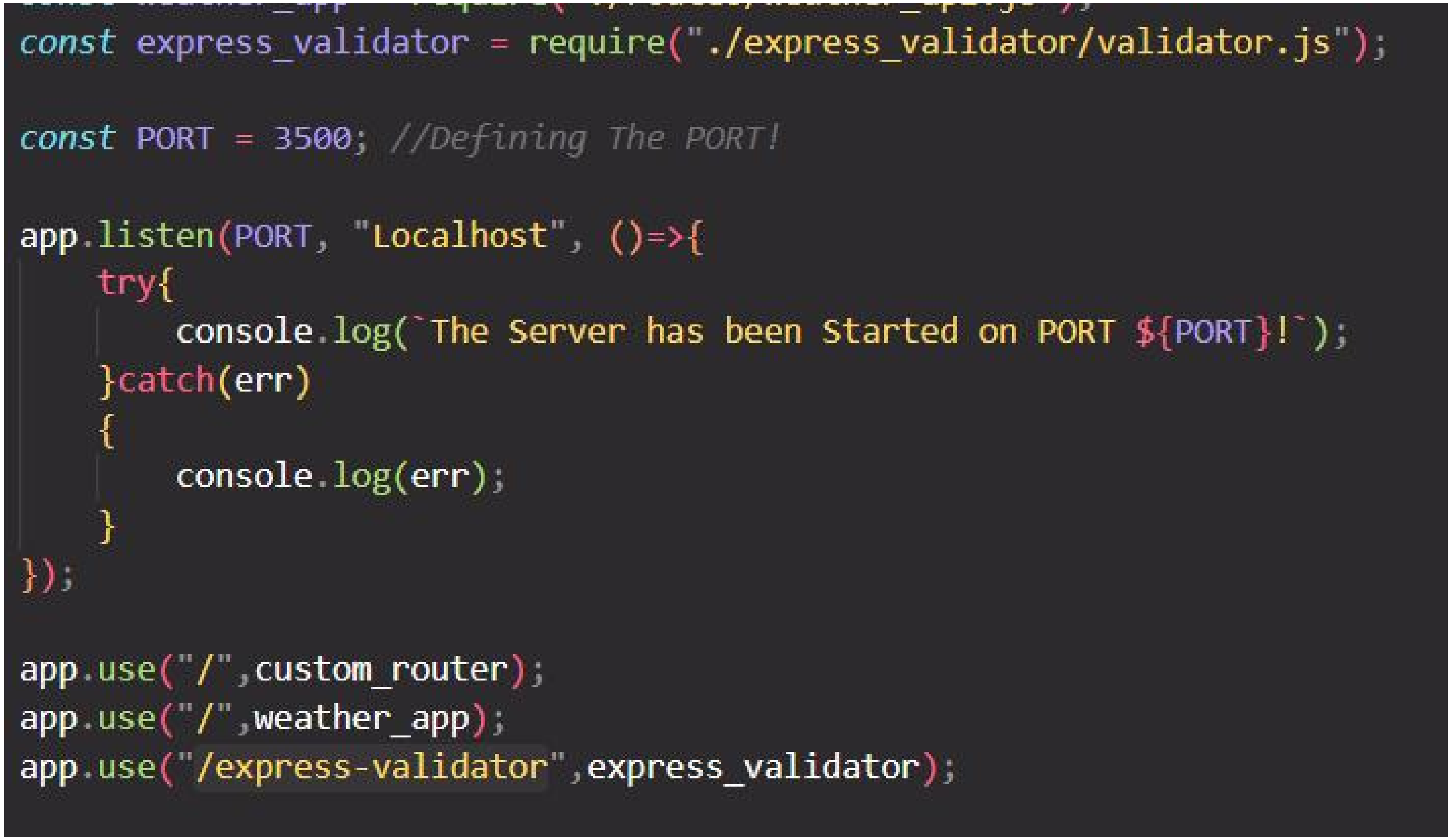
```
const express = require('express');
const { body, validationResult } = require('express-validator');  218.1k (gzipped: 68.5k)
const body_parser = require('body-parser');  503.7k (gzipped: 214.6k)
const path = require('path');

const app  = express();

app.use(body_parser.urlencoded({extended:false}))

app.get("/express-validator",(req,res)=>{
    res.sendFile(path.join(__dirname,'../static/express_validator.html'));
});


app.post('/saveData',
    body('email')
        .isEmail()
        .withMessage("Invalid Email!"),

        (req,res) =>{
            const errors = validationResult(req)
            if(!errors.isEmpty())
            {
                res.send("Email is not Valid");
            }else{
                res.send("Email is Valid!!!");
            }
        }
);

module.exports = app;
```

Example : express-validator.html



```
<form method = "post" action  = "saveData">
    <table>
        <center>
            <h1>Express-Validator</h1>
            <tr>
                <td><label>Enter Your Email</label></td>
                <td><label>:</label></td>
                <td><input type = "email" name = "email" placeholder="Enter The Email"/></td>
            </tr>
            <tr>
                <td></td>
                <td>
                    <input type = "submit" name = "submit" value="SUBMIT"/>
                </td>
                <td></td>
            </tr>
        </center>
    </table>
</form>
```

main file : server.js



```
const express_validator = require("./express_validator/validator.js");

const PORT = 3500; //Defining The PORT!

app.listen(PORT, "Localhost", ()=>{
    try{
        console.log(`The Server has been Started on PORT ${PORT}!`);
    }catch(err)
    {
        console.log(err);
    }
});

app.use("/",custom_router);
app.use("/",weather_app);
app.use("/express-validator",express_validator);
```

Template Engine

- One of the key features of Express is its ability to integrate with template engines, allowing developers to dynamically generate HTML pages with data from their server.

- Pre-requisite :
  - Nodejs and NPM
  - ExpressJS

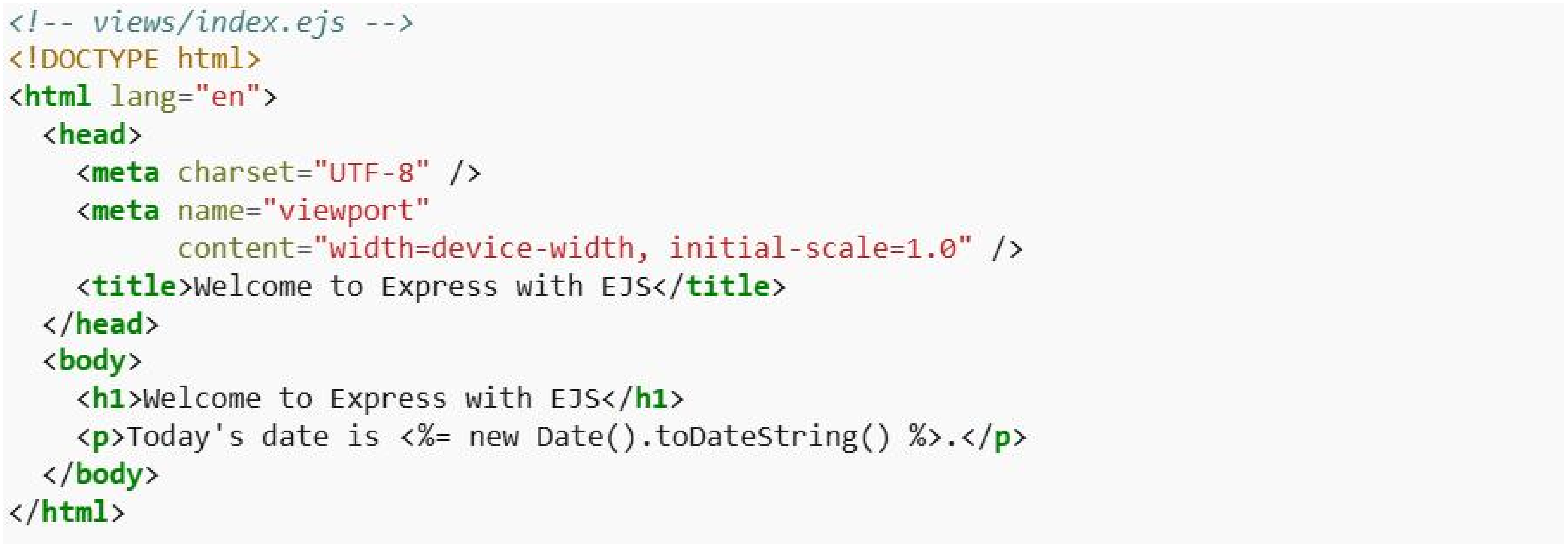
Steps to implement Template engine using express

- Install express.js :
  - command : npm i express ejs
- Set up Express App :
  - const express = require('express');
  - const app = express();
  - const PORT = 3000;
- Configure Express to Use Ejs :
  - app.set(‘view_engine’,’ejs’);
  - app.set(‘views’,__dirname + ‘/views’)

Steps to implement Template engine using express

- Create a Simple EJS Template
    - Create a directory named `views` in your project root directory. Inside the `views` directory, create a new file named `index.ejs` and write a simple EJS template.
- Set Up a Route to Render the EJS Template:
    - app.get('/', (req, res) => {
    - res.render('index');
    - });
- Start the Express Server:
    - app.listen(PORT, () => {
    - console.log(`Server is running on http://localhost:${PORT}`);
    - });

index.ejs



```html
<!-- views/index.ejs -->
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport"
          content="width=device-width, initial-scale=1.0" />
    <title>Welcome to Express with EJS</title>
  </head>
  <body>
    <h1>Welcome to Express with EJS</h1>
    <p>Today's date is <%= new Date().toDateString() %>.</p>
  </body>
</html>
```

Then Run The Node Application.

Other Template Engine

- EJS
- Pug
- Handlebars
- Mustache
- Nunjucks

Thank you