Backend

By Fred Bolder

Axios

Install

npm install axios

Basic get request

```
import axios from 'axios';

axios
    .get('https://www.google.com')
    .then((response) => {
        console.log(response.data);
    })
    .catch((error) => {
        console.error('Error fetching data:', error);
    });
```

Basic get request using async/await

```
import axios from 'axios';

async function fetchData() {
    try {
        const response = await axios.get('https://www.google.com');
        console.log(response.data);
    } catch (error) {
        console.error('Error fetching data:', error);
    }
}

fetchData();
```

Cookies

server.js

```
import express from 'express';
import dotenv from 'dotenv';
import jwt from 'jsonwebtoken';
import cookieParser from 'cookie-parser';
import cors from 'cors';

const app = express();

dotenv.config();

app.use(cors());
app.use(cookieParser());
app.use(express.json());
app.use(express.urlencoded({ extended: true }));

app.get('/create', (req, res, next) => {
    // For test in browser
    const token = jwt.sign({ user: 'Test' }, process.env.SECRET_KEY, { expiresIn: 3600 });
    res.cookie('token', token, {
        expires: new Date(Date.now() + 3600000),
        httpOnly: true,
        secure: true
    });
    res.send("Cookie created");
});

app.post('/login', (req, res, next) => {
    const { email, password } = req.body;
    // Of course this is not how you normally should check the password.
    if (email === 'test@mail.com' && password === '123') {
        const token = jwt.sign({ user: 'Test' }, process.env.SECRET_KEY, { expiresIn: 3600 });
        res.cookie('token', token, {
            expires: new Date(Date.now() + 3600000),
            httpOnly: true,
            secure: true
        });
        res.json({
            message: 'Logged in successfully',
        });
    }
    else {
        res.status(401).send('Unauthorized');
    }
});

app.get('/main', (req, res, next) => {
    try {
        const token = req.cookies['token'];
        const payload = jwt.verify(token, process.env.SECRET_KEY);
        res.send(`Welcome ${payload.user}`);
    } catch (error) {
        res.status(401).send('Unauthorized');
    }
});

app.listen(process.env.PORT, () => {
    console.log(`Server running at port ${process.env.PORT}`);
```

```
});
```

.env

```
PORT=5000
SECRET_KEY=VerySecret!
```

CORS

Cross Origin Resource Sharing
https://expressjs.com/en/resources/middleware/cors.html

Install

npm i cors

Example 1 (all origins)

```
import cors from "cors";

app.use(cors());
```

Example 2 (one origin)

```
import cors from "cors";

app.use(corse({origin: "https://mysite.com"}));
```

Environment variables

Install

npm install dotenv

.gitignore
node_modules
.env

.env (in same folder as package.json)

MYVAR1=Test1
MYVAR2=Test2

script.js

```
import dotenv from "dotenv";

// Read environment variables from .env
dotenv.config();

console.log(process.env.MYVAR1);
console.log(process.env.MYVAR2);
```

Web server

node

Install

npm init -y
in package.json add "type": "module" under "name"

server.js

```
import http from 'http';
const server = http.createServer();

server.addListener('request', (req, res) => {
    res.writeHead(200, { 'Content-Type': 'text/html' });
    res.write('<h1>Hello World!</h1>');
    res.end();
});

server.listen(3000, '127.0.0.1', () => {
    console.log("The server is listening.");
});
```

express

Install

npm init -y
npm i express
in package.json add "type": "module" under "name"

server.js

```
import express from 'express';
const app = express();
const PORT = 3000;

app.get('/', (req, res) => {
    res.send('Hello World!');
});

app.listen(PORT, () => {
    console.log(`The server is listening on port ${PORT}.`);
});
```

express with URL parameter

```
import express from 'express';
const app = express();
const PORT = 3000;

app.get('/products/:productName', (req, res) => {
    res.send(`<h1>Product ${req.params.productName}</h1>`);
});

app.listen(PORT, () => {
    console.log(`The server is listening on port ${PORT}.`);
});

// In browser: http://127.0.0.1:3000/products/Jacket
```

express with query parameter

```
import express from 'express';
const app = express();
const PORT = 3000;

app.get('/products', (req, res) => {
    res.send(`<h1>Product ${req.query.productName}</h1>`);
});

app.listen(PORT, () => {
    console.log(`The server is listening on port ${PORT}.`);
});

// In browser: http://127.0.0.1:3000/products?productName=Shirt
```

express page not found

```
import express from 'express';
const app = express();
const PORT = 3000;

app.get('/', (req, res) => {
    res.send('Hello World!');
});

// Must be after all routings
app.use((req, res) => {
    res.status(404).send("Page not found");
});

app.listen(PORT, () => {
    console.log(`The server is listening on port ${PORT}.`);
});
```

express post

index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Client</title>
  </head>
  <body>
    <h1>Client</h1>
    <form action="http://127.0.0.1:3000/" method="post">
      <input name="data" />
      <button>Submit</button>
    </form>
  </body>
</html>
```

server.js

```
import express from 'express';
const app = express();
const PORT = 3000;

// Makes using req.body possible
app.use(express.urlencoded({ extended: true }));

app.post('/', (req, res) => {
    res.send(`Data ${req.body.data} received.`);
});

app.listen(PORT, () => {
    console.log(`The server is listening on port ${PORT}.`);
});
```

express sendFile

```
import express from 'express';
import path from 'path';
const app = express();
const PORT = 3001;
const fileIndexHtml = path.resolve("./public/index.html");

app.get("/", (req, res) => {
    // Absolute path needed
    res.sendFile(fileIndexHtml);
});

app.listen(PORT, () => {
    console.log(`The server is listening on port ${PORT}.`);
});
```

express - middleware

```
// Parse incoming JSON requests and puts the parsed data in req.body
app.use(express.json());
```

express - your own middleware

Example 1

```
import express from 'express';
const app = express();
const PORT = 3000;

function myMiddleware(req, res, next) {
    req.myData = "Test";
    next();
}

app.use(myMiddleware);

app.get('/', (req, res) => {
    res.send(req.myData);
});

app.listen(PORT, () => {
    console.log(`The server is listening on port ${PORT}.`);
});
```

Example 2

```
import express from 'express';
const app = express();
const PORT = 3000;

function myMiddleware(req, res, next) {
    res.on('finish', () => {
        console.log(`${req.method} ${req.url} ${res.statusCode}`);
    })
    next();
}

app.use(myMiddleware);

app.get('/', (req, res) => {
    res.send("Hello");
});

app.listen(PORT, () => {
    console.log(`The server is listening on port ${PORT}.`);
});
```

Example 3

```
import express from 'express';
const app = express();
const PORT = 3000;

function myMiddleware(req, res, next) {
    console.log("Middleware for About");
    next();
}

app.get('/', (req, res) => {
    res.send("Home");
});

app.get('/about', myMiddleware, (req, res) => {
    res.send("About");
});

app.listen(PORT, () => {
    console.log(`The server is listening on port ${PORT}.`);
});
```

express - normal server setup

controllers/booksControllers.js

```
export const getBooks = (req, res) => {
    res.send("Get books");
};

export const postBook = (req, res) => {
    res.send("Post book");
};

export const getBook = (req, res) => {
    res.send("Get book with ID: " + req.params.ID);
};

export const updateBook = (req, res) => {
    res.send("Update book with ID: " + req.params.ID);
};
```

routes/booksRoutes.js

```
import express from "express";

import { getBooks, postBook, getBook, updateBook }
    from "../controllers/booksControllers.js";

const router = express.Router();

// http:localhost:3000/items/books
router.route("/books").get(getBooks).post(postBook);

// http:localhost:3000/items/book/123
router.route("/book/:ID").get(getBook).put(updateBook);

export default router;
```

server.js

```
import express from 'express';
import books from "./routes/booksRoutes.js";

const app = express();
const PORT = 3000;

app.get('/', (req, res) => {
    res.send('ROOT');
});

app.use("/items", books);

app.listen(PORT, () => {
    console.log(`The server is listening on port ${PORT}.`);
});
```

express - error handling

Only synchronous errors are handled by express.
https://expressjs.com/en/guide/error-handling.html

JSON
{ "firstName": "John", "age": "17" }

Example 1

```
import express from 'express';
const app = express();
const PORT = 3000;

app.use(express.json());

app.post('/', (req, res) => {
    const { firstName } = req.body;
    if (!firstName) {
        const err = new Error();
        err.message = "First name is required";
        err.status = 400;
        throw err;
    }
    res.send(`First name ${req.body.firstName} received`);
});

app.listen(PORT, () => {
    console.log(`The server is listening on port ${PORT}.`);
});
```

Example 2 (error middleware)

```
import express from 'express';
const app = express();
const PORT = 3000;

app.use(express.json());

app.post('/', (req, res, next) => {
    try {
        const { firstName } = req.body;
        if (!firstName) {
            const err = new Error();
            err.message = "First name is required";
            err.status = 400;
            throw err;
        }
        res.send(`First name ${req.body.firstName} received`);
    } catch (error) {
        // next with parameter calls error middleware
        // Do NOT use next("router");
        next(error);
    }
});
```

```
app.use((err, req, res, next) => {
    // An error middleware must have 4 parameters
    // err contains the data from the next parameter
    res.send(err.message);
    // next with param can be used for a second error handler
});

app.listen(PORT, () => {
    console.log(`The server is listening on port ${PORT}.`);
});
```

Sending response

Example 2 (send status)

```
res.sendStatus(404);
```

Example 3 (send status and message)

```
res.status(400).send("This is a message");
```

Example 1 (send status and data)

```
const cars = [
    { id: 1, name: "Volvo" },
    { id: 2, name: "BMW" },
    { id: 3, name: "Audi" },
];

res.status(200).json(cars);
```

lowDB

https://www.npmjs.com/package/lowdb

Install

npm i express lowdb

server.js

```
import express from "express";

const app = express();

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

const port = process.env.PORT || 5000;

// ----- Setup lowDB -----

import { join, dirname } from "node:path";
import { fileURLToPath } from "node:url";

import { Low } from "lowdb";
import { JSONFile } from "lowdb/node";

// db.json file path
const __dirname = dirname(fileURLToPath(import.meta.url));
const file = join(__dirname, "db.json");

// Configure lowdb to write data to JSON file
const adapter = new JSONFile(file);
const defaultData = { records: [] };
const db = new Low(adapter, defaultData);

// Read data from JSON file, this will set db.data content
// If JSON file doesn't exist, defaultData is used instead
await db.read();

// ----- End Setup lowDB -----

//CRUD - Create, Read, Update, Delete

// Create
app.post("/low", async (req, res) => {
    const { records } = db.data;
    // It is not good to use the date for an id, but for this
    // test it is ok.
    records.push({ ...req.body, id: Date.now().toString() });
    await db.write();
    res.status(200).send(records);
});

// Read
app.get("/low", (req, res) => {
    const { records } = db.data;
    res.status(200).send(records);
});

// Update
app.put("/:id", async (req, res) => {
```

```
    const { records } = db.data;
    // I don't think that this await is needed
    let record = await records.find((obj) => obj.id === req.params.id);
    const { title, artist, year } = req.body;
    record.title = title;
    record.artist = artist;
    record.year = year;
    await db.write();
    res.status(200).send(record);
});

// Delete
app.delete("/:id", async (req, res) => {
    const { records } = db.data;
    const removeindex = records.findIndex((item) => item.id === req.params.id);

    if (removeindex != -1) {
        records.splice(removeindex, 1);
    }

    await db.write();
    res.status(200).send(db.data);
});

app.listen(port, () => {
    console.log(`Server is running on ${port}`);
});
```

Unique ID

```
function generateID(data) {
  let id = -1;

    let maxID = data.reduce((acc, current) => {
        let value = parseInt(current.id);
        if (value === NaN) {
            value = 0;
        }
        return value > acc ? value : acc;
    }, 0);

    if (maxID < Number.MAX_SAFE_INTEGER) {
        id = maxID + 1;
    } else {
        // Search unused ID
        const sortedIDs = [];
        for (let i = 0; i < data.length; i++) {
            let value = parseInt(data[i].id);
            if (value === NaN) {
                value = 0;
            }
            sortedIDs.push(value);
        }
        sortedIDs.sort((a, b) => a - b);
        console.log(sortedIDs);
        id = 0;
        let idx = 0;
        let prev = 0;
        while (idx < sortedIDs.length && id === 0) {
          if ((sortedIDs[idx] - prev) > 1) {
           id = sortedIDs[idx] - 1;
          }
          prev = sortedIDs[idx];
          idx++;
        }
    }
    return id.toString();
}
```

Uploading a file

install

npm i express multer dotenv

index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Upload file</title>
  </head>
  <body>
    <form action="/" method="post" enctype="multipart/form-data">
      <input type="file" name="avatar" />
      <input type="submit" />
    </form>
  </body>
</html>
```

server.js

```
import express from 'express';
import dotenv from "dotenv";
import multer from 'multer';
import path from "path";
import {fileURLToPath} from 'url';

dotenv.config();

const storage = multer.diskStorage({
    destination: function (req, file, cb) {
        cb(null, './');
    },
    filename: function (req, file, cb) {
        cb(null, file.originalname);
    }
});

const upload = multer({ storage: storage });

const app = express();

const __filename = fileURLToPath(import.meta.url);
const __dirname = path.dirname(__filename);
app.use(express.static(path.join(__dirname, './')));

const PORT = process.env.PORT || 4000;

app.post('/', upload.single('avatar'), function (req, res, next) {
    // req.file is the `avatar` file
    // req.body will hold the text fields, if there were any
    res.send("File uploaded");
})

app.get('/', (req, res) => {
    //const __dirname = process.cwd();
    res.sendFile(path.join(__dirname, 'index.html'));
```

```
});

app.listen(PORT, () => {
    console.log(`The server is listening on port ${PORT}.`);
});
```

MongoDB

https://www.mongodb.com/docs/manual/tutorial/install-mongodb-on-ubuntu/
https://www.mongodb.com/docs/manual/tutorial/query-arrays/
https://www.mongodb.com/docs/manual/tutorial/query-array-of-documents/
Default port: 27017
mongodb://localhost:27017
mongodb-compass://localhost:27017

Ubuntu version
lsb_release -d

Start MongoDB
sudo systemctl start mongod

Check status
sudo systemctl status mongod

Stop MongoDB
sudo systemctl stop mongod

Use MongoDB
mongosh
or
mongodb-compass

Command examples

Create or switch to database
use mydb

Show databases (empty databases are not shown)
show dbs

Create collection
db.createCollection("users")

Show collections
show collections

Insert a document into the collection
db.users.insertOne({ firstName: "John", lastName: "Smith", age: 30})

Show all documents in a nice way
db.users.find().pretty()

Show all document except the first 3 documents
db.users.find().skip(3)

Show only the third document
db.users.find().skip(2).limit(1)

Show everything except age
```
db.users.find(
  {},
  { age: 0 }
)
```

Show only age and id (id is normally always shown, but you can hide it)
```
db.users.find(
  {},
  { age: 1 }
```

)

Show users with an age greater than 30
db.users.find({age: {$gt: 30}}).pretty()

Show users with an age greater than 20 and less than 30
db.info.find({$and: [{age: {$gt: 20}}, {age: {$lt: 30}}]}).pretty()

Show users with an age of 12 or 30
db.users.find({age: {$in: [12, 30]}}).pretty()

Show users that have not an age of 12 or 30
db.users.find({age: {$nin: [12, 30]}}).pretty()

Search in the array grades that has objects with a score property
db.saloon.find({grades : { $elemMatch:{"score":{$gt : 90}}}});
https://www.mongodb.com/docs/manual/reference/operator/query/elemMatch/

Import
must be executes in the terminal, not in mongosh!
mongoimport --db=meal --collection=saloon --file=saloon.json

Express with MongoDB

```
import express from 'express'
import dotenv from'dotenv'
import { MongoClient, ObjectId } from 'mongodb'


const app = express()

//Loading the environment variables from env

dotenv.config()

const PORT = process.env.PORT || 4000
const DB_URL= process.env.DB_URL

const dbName = 'usersDatabase'
const collectionName = 'users'

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

app.use(express.static('public'))

app.get('/', (req, res) =>{
    res.sendFile()
})

/*
first we decide the database where to store the data
by use syntax
then decide about the collection where to store the data
db.collectionName.insertOne()
db.collectionName.find()

*/

// Storing / creating data in database
app.post('/', async (req, res)=>{
    const {firstName, lastName} = req.body
    const user = {
        firstName,
        lastName
    }

    const client = new MongoClient(DB_URL)

    try{
        const targetDatabase = client.db(dbName)
        const collection = targetDatabase.collection(collectionName)
        const result = await collection.insertOne(user)
        console.log(result)
    }
    catch(err){
        console.log(err);
    }
    finally{
        client.close()
    }
    res.redirect('/')
```

```
})

// Reading Data from database
app.get('/get-users', async (req, res)=>{
    const client = new MongoClient(DB_URL)
    try{
        const targetDatabase = client.db(dbName)
        const collection = targetDatabase.collection(collectionName)
        const result = await collection.find().toArray()
        console.log(result);

        //To have proper html text

        const htmlText = `
            ${result.map(user => '<p>'+user.firstName+' '+user.lastName+'</p>'+user._id).join('')}
        `
        res.send(htmlText)
    }
    catch(err){
        console.log(err);
    }
    finally{
        client.close()
    }

})

app.delete('/delete-user/:firstName', async (req, res) =>{
    const {firstName} = req.params


    const client = new MongoClient(DB_URL)
    try{
        const targetDatabase = client.db(dbName)
        const collection = targetDatabase.collection(collectionName)
        const result = await collection.deleteOne({firstName:firstName})

        console.log(result)
        if(result.deletedCount) res.send('user deleted')
        else res.send('Some thing went wrong')
    }
    catch(err){
        console.log(err);
    }
    finally{
        client.close()
    }

})

app.put('/update-user', async (req, res) =>{

    const {firstName,lastName, updatedFirstName} = req.body

    const client = new MongoClient(DBt_URL)
    try{
        const targetDatabase = client.db(dbName)
        const collection = targetDatabase.collection(collectionName)
        const result = await collection.updateOne({firstName: firstName},{$set:{firstName: updatedFirstName}})

        console.log(result)
        if(result.modifiedCount) res.send('user updated')
        else res.send('Some thing went wrong')
```

```
    }
    catch(err){
        console.log(err);
    }
    finally{
        client.close()
    }
})

app.listen(PORT, ()=>console.log(`Backend server running at port ${PORT}`))
```

Mongoose

https://www.mongodb.com/atlas/database
https://mongoosejs.com/docs/guides.html

.env
PORT=5000
CONNECTION_URI=

postControllers.js

```
import PostMessageModel from "../models/postMessage.js";

export const getPosts = async (req, res) => {
  try {
    const posts = await PostMessageModel.find();
    res.status(200).json(posts);
  } catch (error) {
    res.status(404).json({ msg: error.message });
  }
};

export const createPost = async (req, res) => {
  try {
    const post = req.body;
    const newPost = new PostMessageModel(post);
    await newPost.save();
    res.status(201).json(newPost);
  } catch (error) {
    res.status(409).json({ msg: error.message });
  }
};

export const updatePost = async (req, res) => {
  try {
    const updatedPost = await PostMessageModel.findByIdAndUpdate(
      req.params.id,
      req.body,
      {
        new: true,
      },
    );
    res.status(201).json(updatedPost);
  } catch (error) {
    res.status(409).json({ msg: error.message });
  }
};

export const deletePost = async (req, res) => {
  try {
    await PostMessageModel.findByIdAndRemove(req.params.id);
    res.status(201).json({ message: "Post deleted!" });
  } catch (error) {
    res.status(409).json({ msg: error.message });
  }
};
```

https://mongoosejs.com/docs/guides.html

postMessage.js

```
import mongoose from "mongoose";

const postSchema = mongoose.Schema({
  title: { type: String, required: true },
  message: String,
  name: String,
  createAt: {
    type: Date,
    default: new Date(),
  },
});

const PostMessageModel = mongoose.model("PostMessageCollection", postSchema);

export default PostMessageModel;
```

postsRoutes.js

```
import express from "express";
import {
  createPost,
  deletePost,
  getPosts,
  updatePost,
} from "../controllers/postControllers.js";

const router = express.Router();

//http://localhost:5000/posts
router.get("/", getPosts);

router.post("/", createPost);

router.patch("/:id", updatePost);

router.delete("/:id", deletePost);

export default router;
```

server.js

```
import express from "express";
import cors from "cors";
import dotenv from "dotenv";

import mongoose from "mongoose";
import postsRoutes from "./routes/postsRoutes.js";

const app = express();
dotenv.config();

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

const PORT = process.env.PORT;

app.get("/", (req, res) => {
  res.send("Hello world!");
});

app.use("/posts", postsRoutes);

mongoose
  .connect(process.env.CONNECTION_URI)
  .then(() => {
    app.listen(PORT, () => {
      console.log(`DB connected and server is running on port ${PORT}`);
    });
  })
  .catch((err) => console.log(err));
```

Authentication

helpers.js

```
import fs from "fs";

const fileName = "data.txt";

export function writeToFileSystem(string) {
  try {
    fs.writeFileSync(fileName, string);
    console.log(`Data written to ${fileName}`);
  } catch (error) {
    console.error("Error writing file (did you pass the correct data?)");
    console.error(error.message);
  }
}

export function readFromFileSystem() {
  try {
    return fs.readFileSync(fileName, { encoding: "utf8", flag: "r" });
  } catch (error) {
    console.error("Error reading file (does it exist?)");
    console.error(error.message);
  }
}
```

register.js

```
import bcrypt from "bcrypt";
import { writeToFileSystem } from "./helpers.js";

const password = process.argv[2];

async function register(password) {
    await bcrypt.hash(password, 12, function (err, hash) {
        writeToFileSystem(hash);
        console.log(hash);
    });
}

register(password);
```

login.js

```
import bcrypt from "bcrypt";
import { readFromFileSystem } from "./helpers.js";

const password = process.argv[2];

async function login(password) {
    let hash = readFromFileSystem();
    await bcrypt.compare(password, hash, function(err, result) {
        if (result) {
          console.log("The password is correct.");
        } else {
            console.log("The password is wrong!");
        }
    })
}

login(password);
```

Publishing on render.com

Important

Cookies do not work on render.com with a separate frontend. You can build the frontend and copy the dist folder (vite) in the backend. The .env files must be ignored. The .env for the backend you define on render.com, but remove the port.
On render.com create an environment variable NODE_VERSION and give it the value of the version of Node that the project is working with (example: 19.7.0).

Backend

server.js

```
app.use(cors({ origin: ['https://games-41ql.onrender.com/'], credentials: true }));
app.use(express.static(path.resolve("./frontend/dist/")));

app.get("/", (req, res) => {
  res.sendFile(path.resolve("./frontend/dist/index.html"));
});
```

generateToken.js

```
import jwt from "jsonwebtoken";

const generateToken = (res, userId) => {
  const token = jwt.sign({ userId }, process.env.JWT_SECRET, {
    expiresIn: "30d",
  });

  res.cookie("jwt", token, {
    httpOnly: true,
    //secure: process.env.NODE_ENV !== "development",
    secure: true,
    sameSite: "lax",
    maxAge: 30 * 24 * 60 * 60 * 1000,
  });
};

export default generateToken;
```

Frontend

```
await axios.post(
  `${import.meta.env.VITE_BE_URL}/api/users/bal`,
  { level: level },
  { withCredentials: true }
);
```