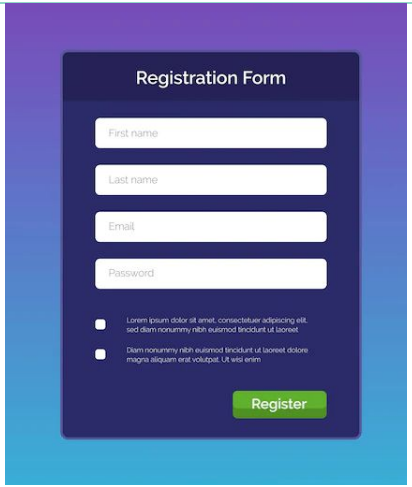
Express Middleware, Forms and Saving Data in FS

All functionality b/w req & res



Registration Form

First name

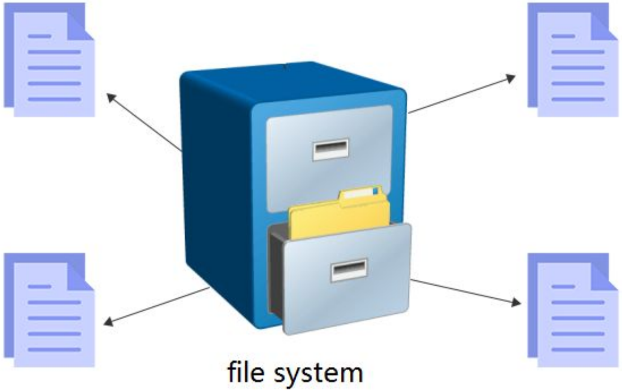
Last name

Email

Password

□ Lorem ipsum dolor sit amet. consectetuer adipiscing elit. sed diam nonummy nibh euismod tincidunt ut laoreet

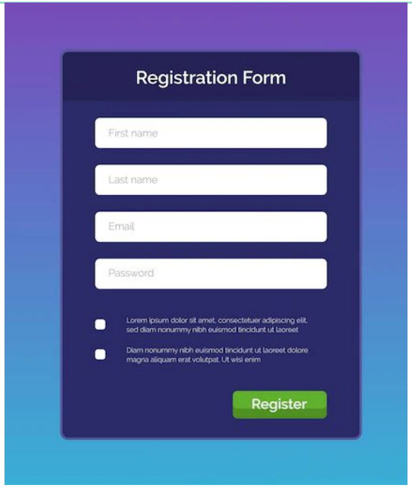
□ Diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim



Express Middleware, Forms and Saving Data in FS

Reading req & process further

By Aquib Ajani



Registration Form

First name

Last name

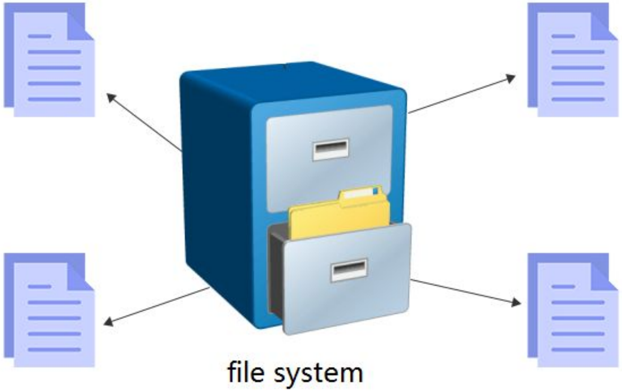
Email

Password

□ Lorem ipsum dolor sit amet. consectetuer adipiscing elit. sed diam nonummy nibh euismod tincidunt ut laoreet

□ Diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim

Register



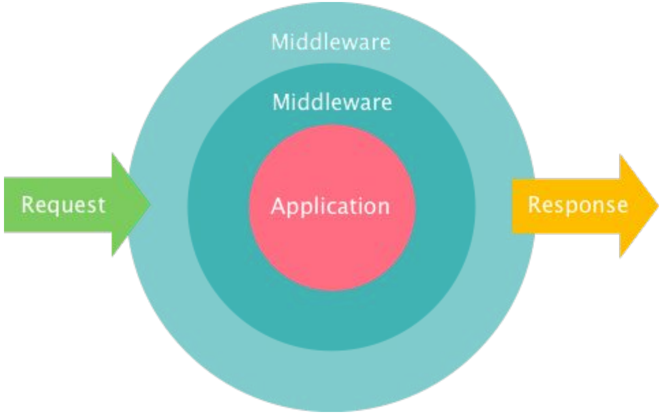
Express Middleware

What is middleware?

Middleware functions have access to the request object (req), the response object (res), and the next middleware function to be invoked in the application's request-response cycle.

Middleware functions can perform the following tasks

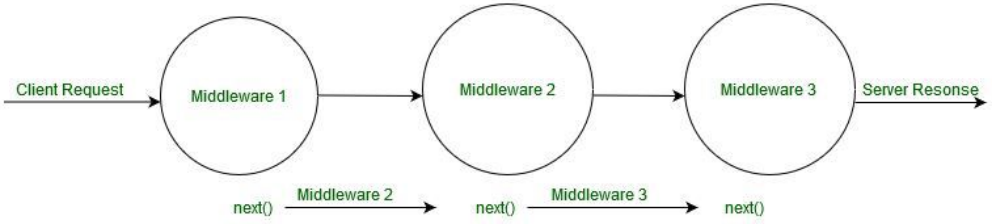
- Execute a piece of code
- Modify the request and the response objects
- Terminate the request-response cycle
- Invoke the next middleware function



Express Middleware

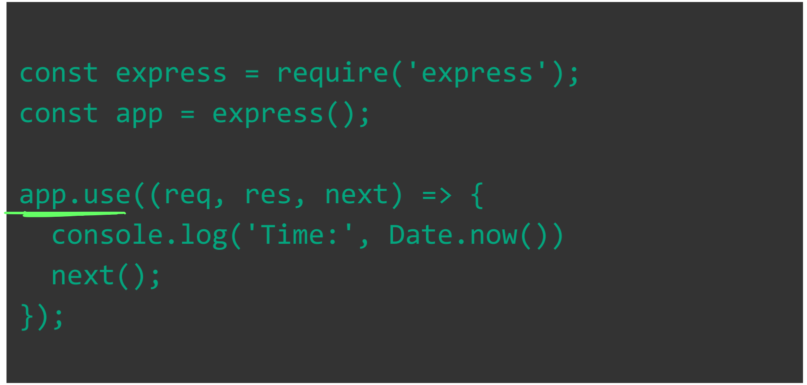
Types of Express Middleware

- Application-level middleware
- Router-level middleware
- Error-handling middleware
- Built-in middleware
- Third-party middleware



Application-level middleware

⇒ As soon as req is sent or res is received at that time which middleware invoke automatically without requiring any -(path/Route



```
const express = require('express');
const app = express();

app.use((req, res, next) => {
  console.log('Time:', Date.now())
  next();
});
```

Router-level middleware

⇒ Runs after the application is loaded and before anything is shown at the screen.

```
const express = require('express');
const app = express();
const router = express.Router();

router.use((req, res, next) => {
  console.log('Time: ', Date.now());
  next();
})

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

router.get('/about', (req, res) => {
  res.send('About page);
})

app.use('/', router);
```

Error-handling middleware

⇒ Used to display errors

```
app.use((err, req, res, next) => {
  console.error(err.stack);
  res.status(500).send('Something went wrong!');
});
```

Built-in middleware

```
express.static serves static assets such as HTML files, images, and so on.
express.json parses incoming requests with JSON payloads
express.urlencoded parses incoming requests with URL-encoded payloads
```

Text → JSON

submit sth

Third-party middleware → External Packages built for Express

```
npm install cookie-parser
```

```
const express = require('express');
const app = express();
const cookieParser = require('cookie-parser');

// load the cookie-parsing middleware
app.use(cookieParser());

app.get('/', function (req, res) {
    console.log('Cookies: ', req.cookies);
    console.log('Signed Cookies: ', req.signedCookies);
});
```

Express Middleware, Forms and Saving Data in FS

search

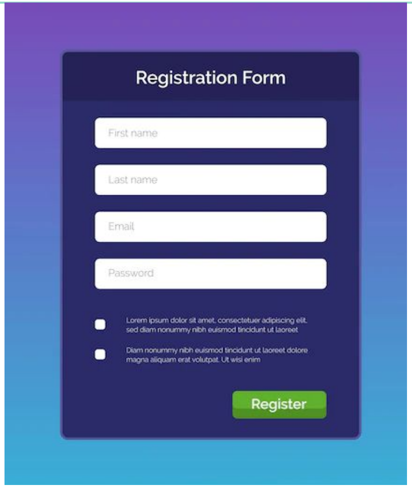
Login, Sign-up forms

Payment details,

credit-debit card info

LinkedIn forms

Google Search



Registration Form

First name

Last name

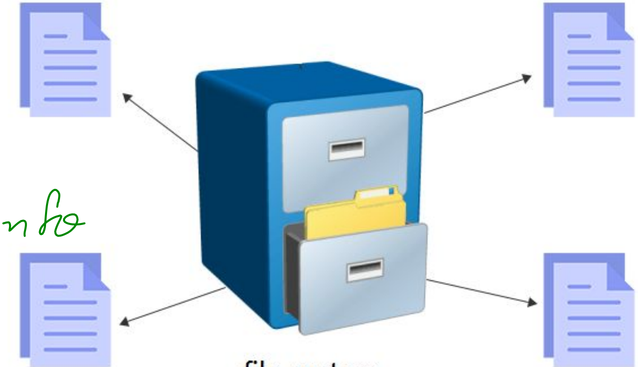
Email

Password

□ Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet

□ Diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim

Register



file system

Handling forms in Express

Step 1. Initialise the project with **npm init**

```
npm init
```

Step 2. Install **express**

```
npm install express
```

Step 3. Create the **html form**

.html

```
<form method="POST" action="/submit-form">
  <input type="text" name="username" />
  <input type="submit" />
</form>
```

Form will go to html file to index.js

Form will be controlled from here

action => "./index.js"

<input name="—">

works like a link b/w Frontend & Back-end

Post => Private Data

Get => Url (Public)

Handling forms in Express

Step 4. Reading the form data in Express (urlencoded)

```
const express = require('express');
const app = express();

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

app.post('/submit-form', (req, res) => {
  const username = req.body.username
  res.end();
});
```

Step 5. Displaying the data to the user

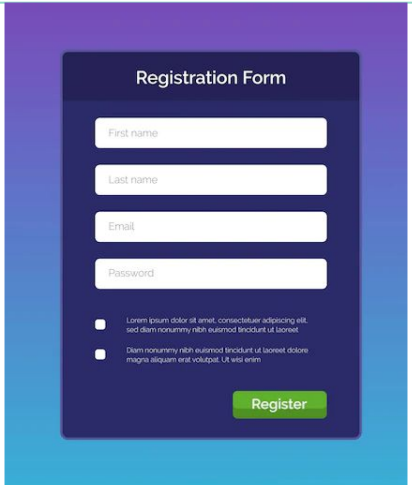
```
document.getElementById(“userName”).innerHTML = username;
```

Code for thought

Try validating the form data with the express-validator package

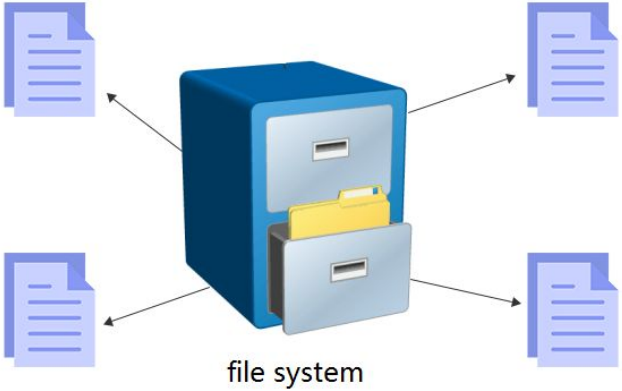
↳ Research by yourself

Express Middleware, Forms and Saving Data in FS



□ Lorem ipsum dolor sit amet. consectetuer adipiscing elit. sed diam nonummy nibh euismod tincidunt ut laoreet

□ Diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim



Node.js File System

To work with files, we have to use the file system module

```
let fs = require('fs');
```

There are 5 major actions we can perform on files :

1. Read data from a file
2. Create a new file
3. Update an existing file
4. Delete an existing file
5. Rename an existing file

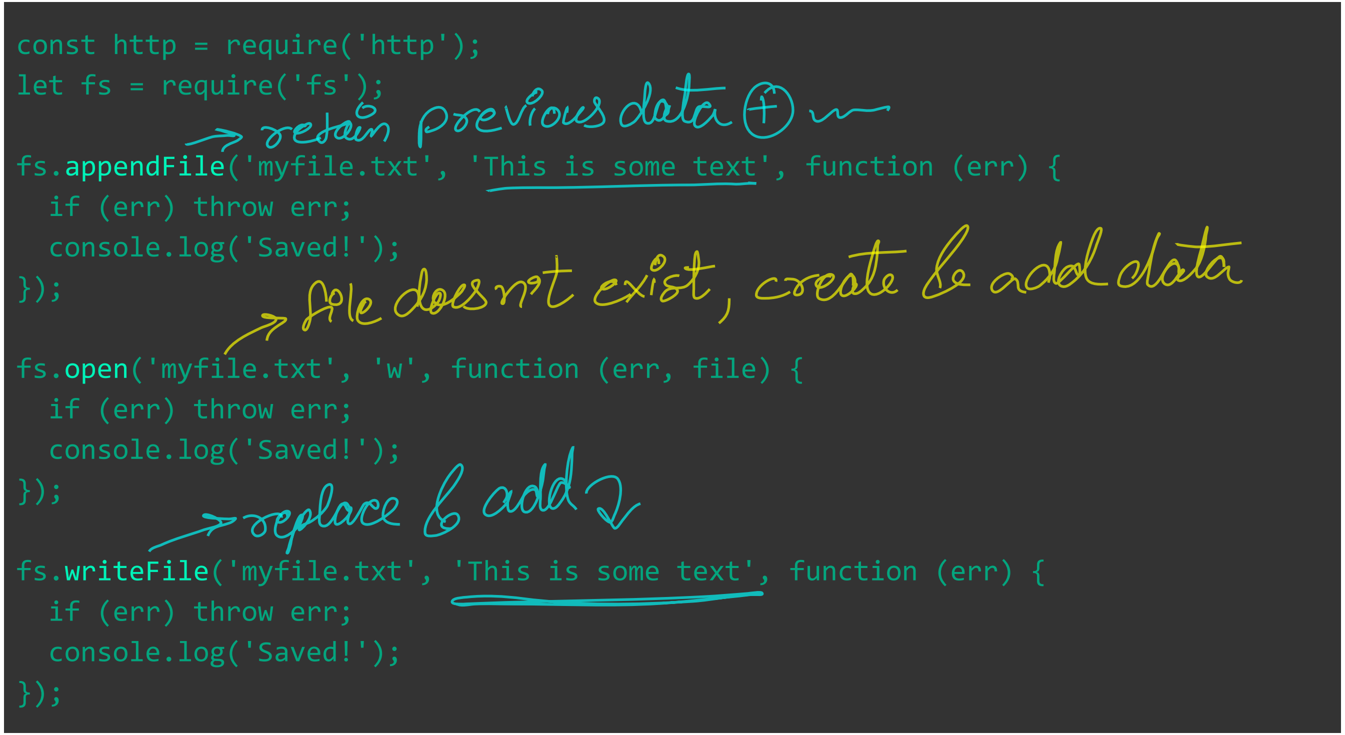
Node.js File System

Read data from a file

```
const http = require('http');
let fs = require('fs');
http.createServer(function (req, res) {
  fs.readFile('myfile.html', function(err, data) {
    res.writeHead(200, {'Content-Type': 'text/html'});
    res.write(data);
    return res.end();
  });
}).listen(3000);
```

Node.js File System

Create a new file



```
const http = require('http');
let fs = require('fs');
```

→ retain previous data ⊕

```
fs.appendFile('myfile.txt', 'This is some text', function (err) {
  if (err) throw err;
  console.log('Saved!');
});
```

→ file does not exist, create & add data

```
fs.open('myfile.txt', 'w', function (err, file) {
  if (err) throw err;
  console.log('Saved!');
});
```

→ replace & add ↓

```
fs.writeFile('myfile.txt', 'This is some text', function (err) {
  if (err) throw err;
  console.log('Saved!');
});
```

Node.js File System

Update an existing file

```
const http = require('http');
let fs = require('fs');

fs.appendFile('myfile.txt', 'This is some text', function (err) {
  if (err) throw err;
  console.log('Saved!');
});

fs.writeFile('myfile.txt', 'This is some text', function (err) {
  if (err) throw err;
  console.log('Saved!');
});
```

Node.js File System

Delete an existing file

```
const http = require('http');
let fs = require('fs');

fs.unlink('myfile.txt', function (err) {
  if (err) throw err;
  console.log('File deleted!');
});
```

Node.js File System

Rename an existing file

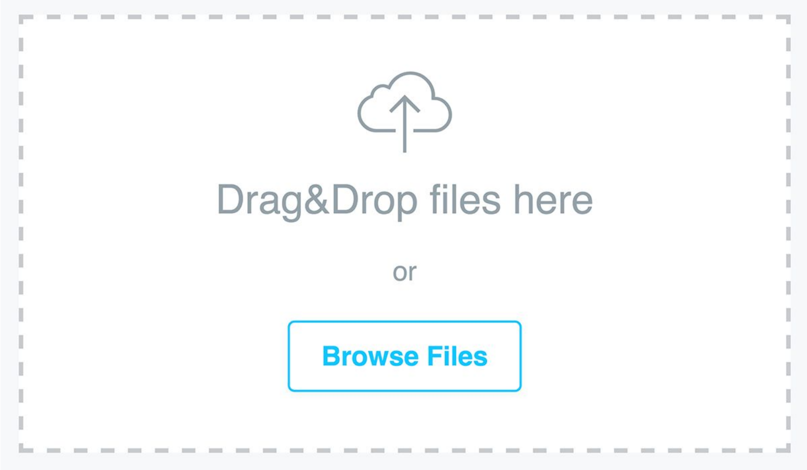
```
const http = require('http');
let fs = require('fs');

fs.rename('myfile.txt', 'mynewfile.txt', function (err) {
  if (err) throw err;
  console.log('File Renamed!');
});
```

new

Code for thought

Ques - 2

Try uploading files with Node.js using the formidable module



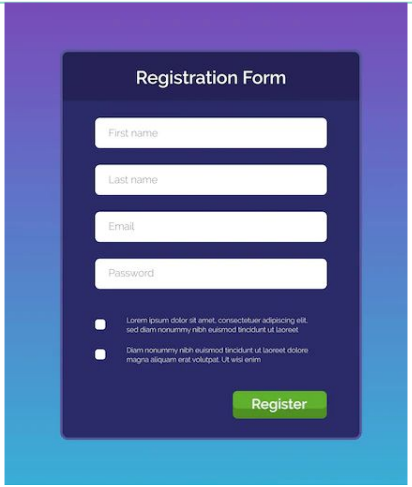
Drag&Drop files here

or

Browse Files

Ques⇒ 3 How to put data when you create file by open method ?

Express Middleware, Forms and Saving Data in FS



Registration Form

First name

Last name

Email

Password

□ Lorem ipsum dolor sit amet. consectetuer adipiscing elit. sed diam nonummy nibh euismod tincidunt ut laoreet

□ Diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim