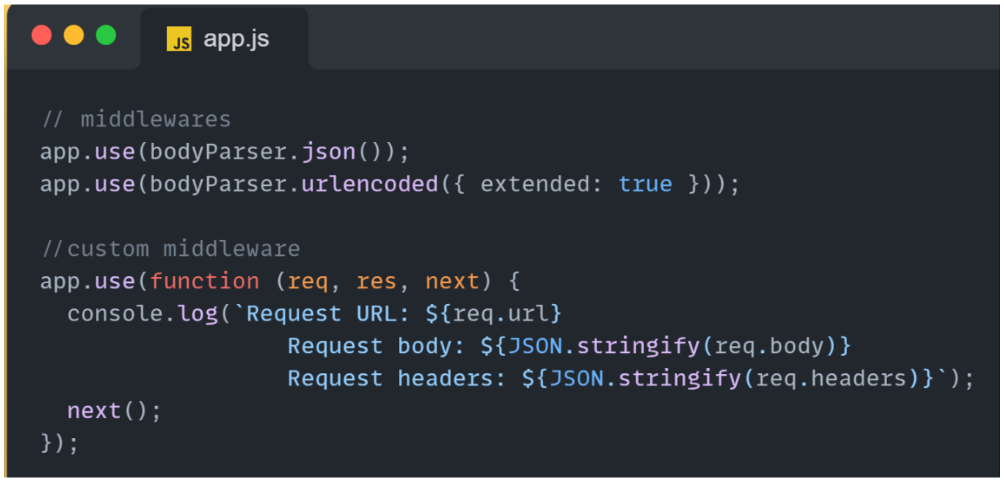
Task 1: Express Middleware Basics

Steps:

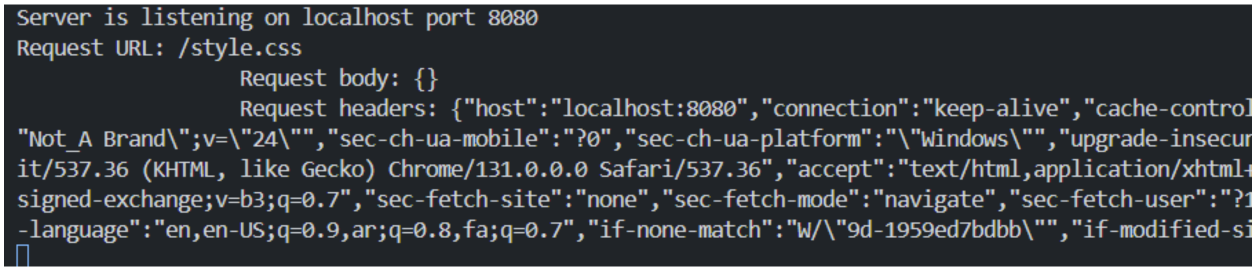
1. Create a simple Express server
2. Add a custom middleware that logs request details
3. Test with different routes



app.js

```
// middlewares
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: true }));

//custom middleware
app.use(function (req, res, next) {
  console.log(`Request URL: ${req.url}
                Request body: ${JSON.stringify(req.body)}
                Request headers: ${JSON.stringify(req.headers)}`);
  next();
});
```



```
Server is listening on localhost port 8080
Request URL: /style.css
                Request body: {}
                Request headers: {"host":"localhost:8080","connection":"keep-alive","cache-control
"Not_A Brand\";v=\"24\"","sec-ch-ua-mobile":"?0","sec-ch-ua-platform":"\"Windows\"","upgrade-insecur
it/537.36 (KHTML, like Gecko) Chrome/131.0.0.0 Safari/537.36","accept":"text/html,application/xhtml+
signed-exchange;v=b3;q=0.7","sec-fetch-site":"none","sec-fetch-mode":"navigate","sec-fetch-user":"?1
-language":"en,en-US;q=0.9,ar;q=0.8,fa;q=0.7","if-none-match":"W/\"9d-1959ed7bdbb\"","if-modified-si
```

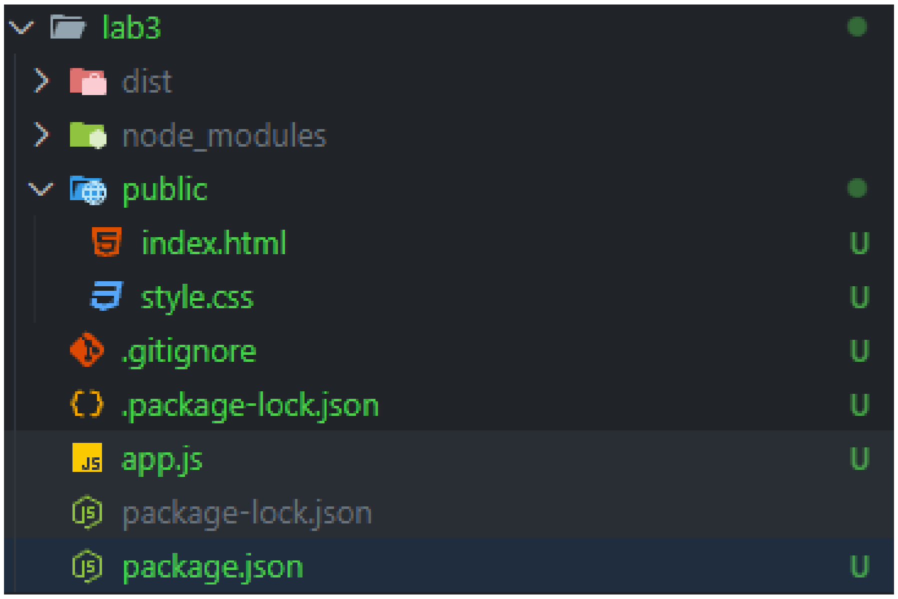
Task 2: Serving Static Files with `express.static()`

Steps:

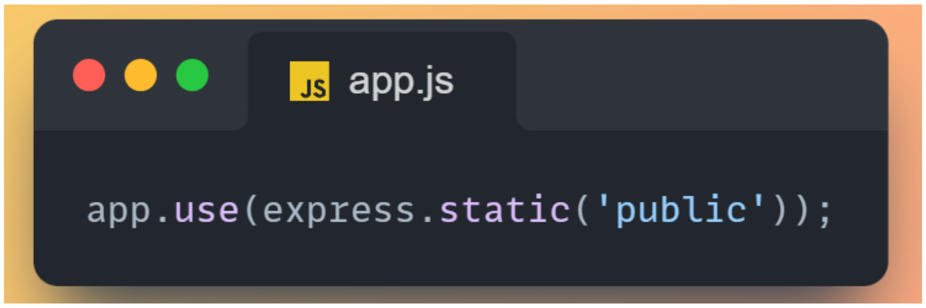
1. Create a `public` folder with `index.html` and `style.css`
2. Serve static files using `express.static()`
3. Test by accessing files in the browser

Folder Structure:

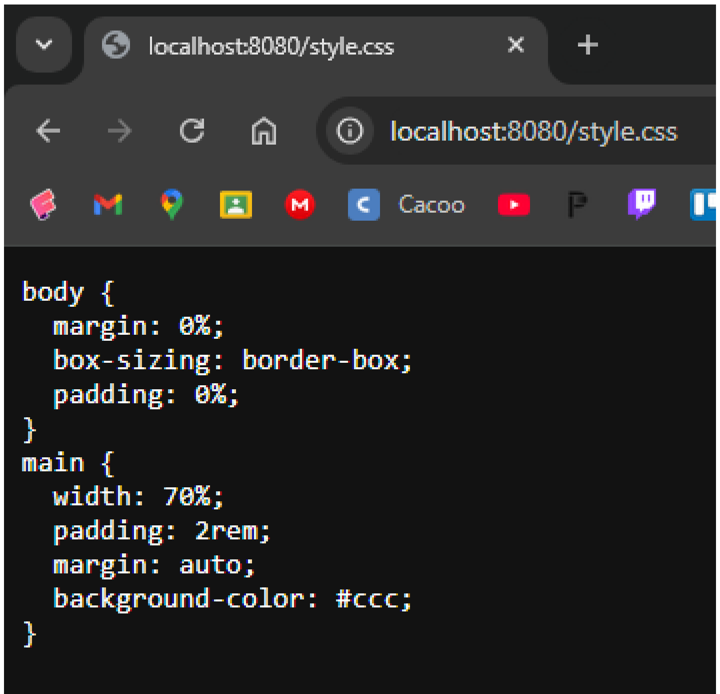
```
/project-folder
|── /public
|    ├── index.html
|    ├── style.css
|── server.js
```



```
lab3
  dist
  node_modules
  public
    index.html        U
    style.css         U
  .gitignore          U
  .package-lock.json  U
  app.js              U
  package-lock.json
  package.json        U
```



```js
app.use(express.static('public'));
```

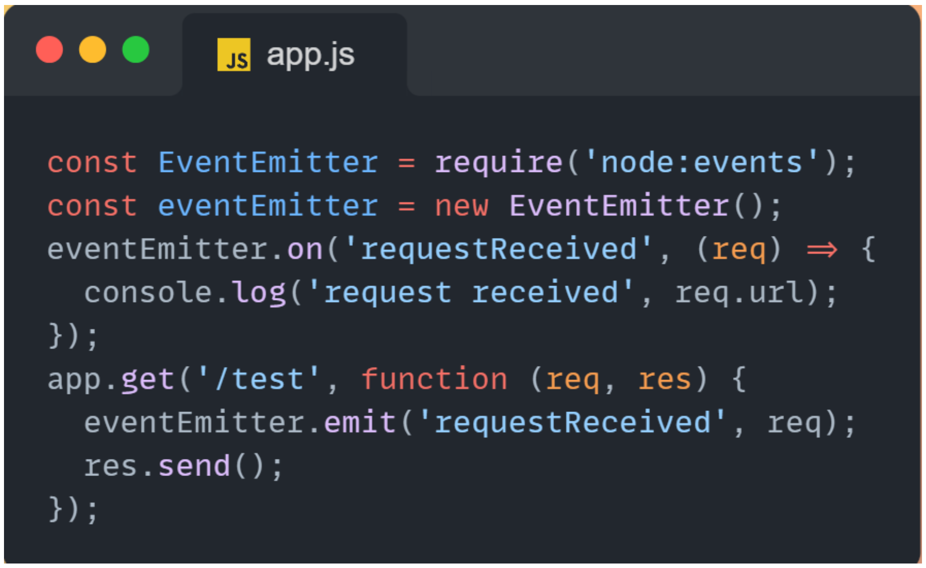


```css
body {
  margin: 0%;
  box-sizing: border-box;
  padding: 0%;
}
main {
  width: 70%;
  padding: 2rem;
  margin: auto;
  background-color: #ccc;
}
```

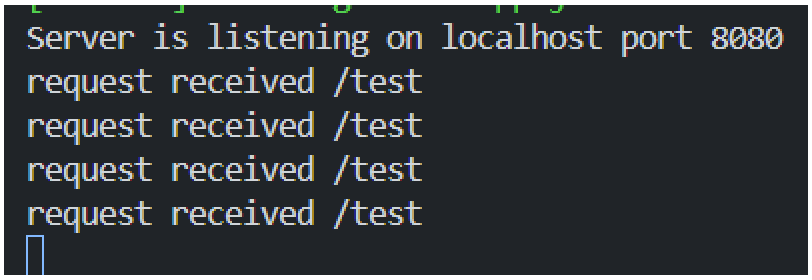
Task 3: Using `EventEmitter` to Log Events

Steps:

1. Create an `EventEmitter` instance
2. Listen for a custom event `requestReceived`
3. Emit the event when a request is made
4.



```javascript
const EventEmitter = require('node:events');
const eventEmitter = new EventEmitter();
eventEmitter.on('requestReceived', (req) => {
  console.log('request received', req.url);
});
app.get('/test', function (req, res) {
  eventEmitter.emit('requestReceived', req);
  res.send();
});
```

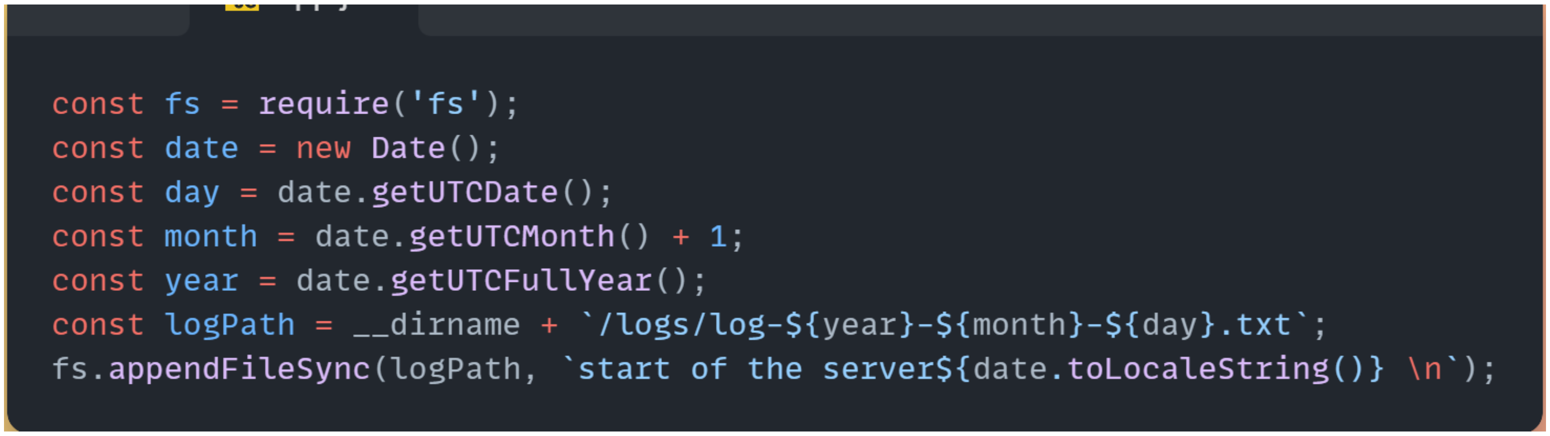


```
Server is listening on localhost port 8080
request received /test
request received /test
request received /test
request received /test
```

Task 4: File System Operations (`fs` module)

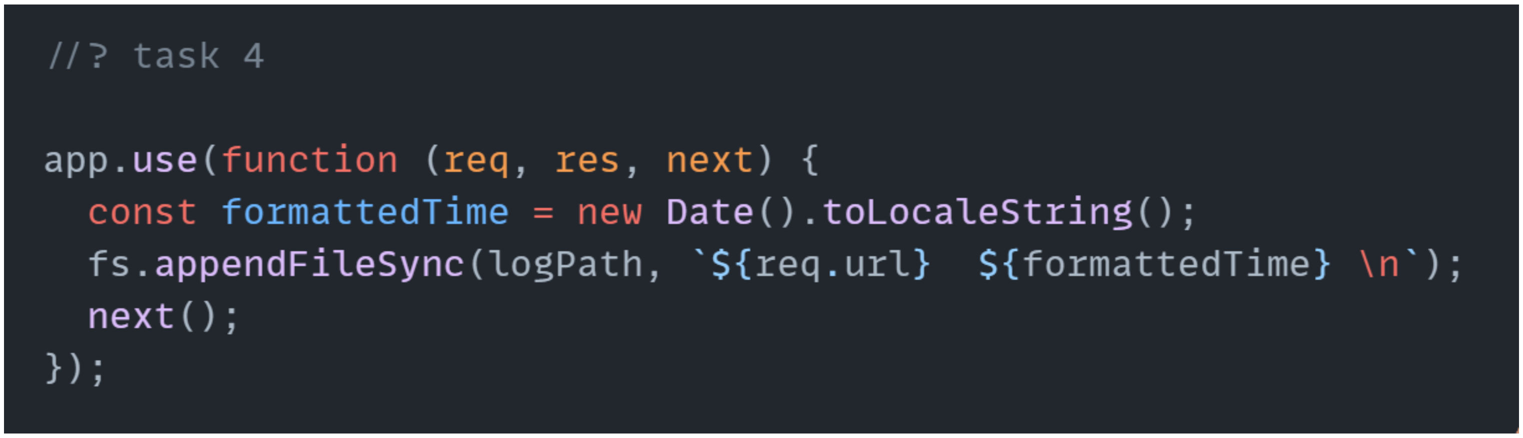
Steps:

1. Use `fs.writeFile()` to log requests into `log.txt`
2. Use `fs.readFile()` to display logs via an API endpoint
3. Test logging by making requests



```javascript
const fs = require('fs');
const date = new Date();
const day = date.getUTCDate();
const month = date.getUTCMonth() + 1;
const year = date.getUTCFullYear();
const logPath = __dirname + `/logs/log-${year}-${month}-${day}.txt`;
fs.appendFileSync(logPath, `start of the server${date.toLocaleString()} \n`);
```

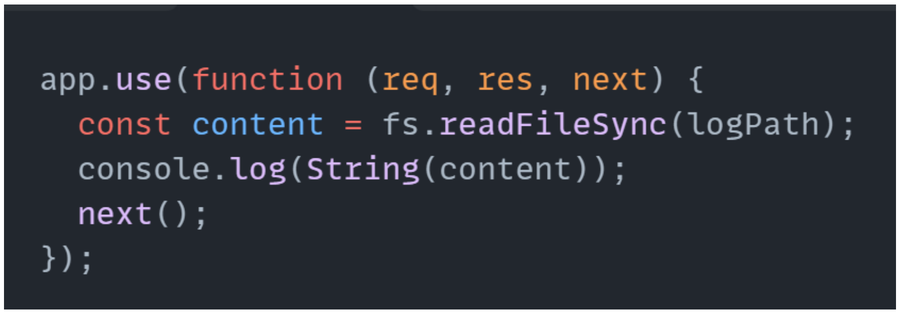
middleware to run on all requests and log them



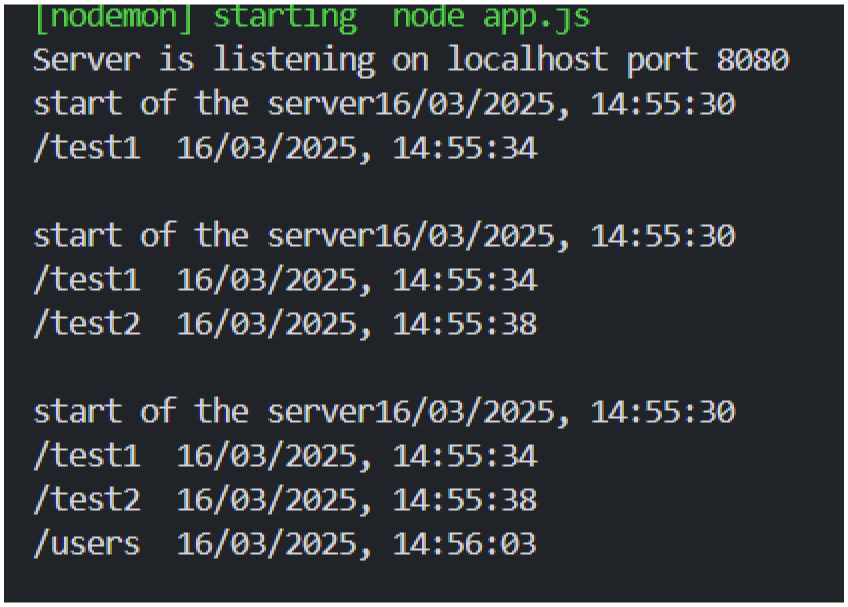
```javascript
//? task 4

app.use(function (req, res, next) {
  const formattedTime = new Date().toLocaleString();
  fs.appendFileSync(logPath, `${req.url}  ${formattedTime} \n`);
  next();
});
```

read them in console



```javascript
app.use(function (req, res, next) {
  const content = fs.readFileSync(logPath);
  console.log(String(content));
  next();
});
```



```
[nodemon] starting `node app.js`
Server is listening on localhost port 8080
start of the server16/03/2025, 14:55:30
/test1  16/03/2025, 14:55:34

start of the server16/03/2025, 14:55:30
/test1  16/03/2025, 14:55:34
/test2  16/03/2025, 14:55:38

start of the server16/03/2025, 14:55:30
/test1  16/03/2025, 14:55:34
/test2  16/03/2025, 14:55:38
/users  16/03/2025, 14:56:03
```