ACADEMIND

“Modern” NodeJS

Utilizing Modern JS Features

Udemy

Don't Take "Modern" Too Literally

The syntax you learnt in this course is 100% correct, 100% up-to-date AND used in the vast majority of projects!

But NodeJS also support more modern JavaScript syntax features

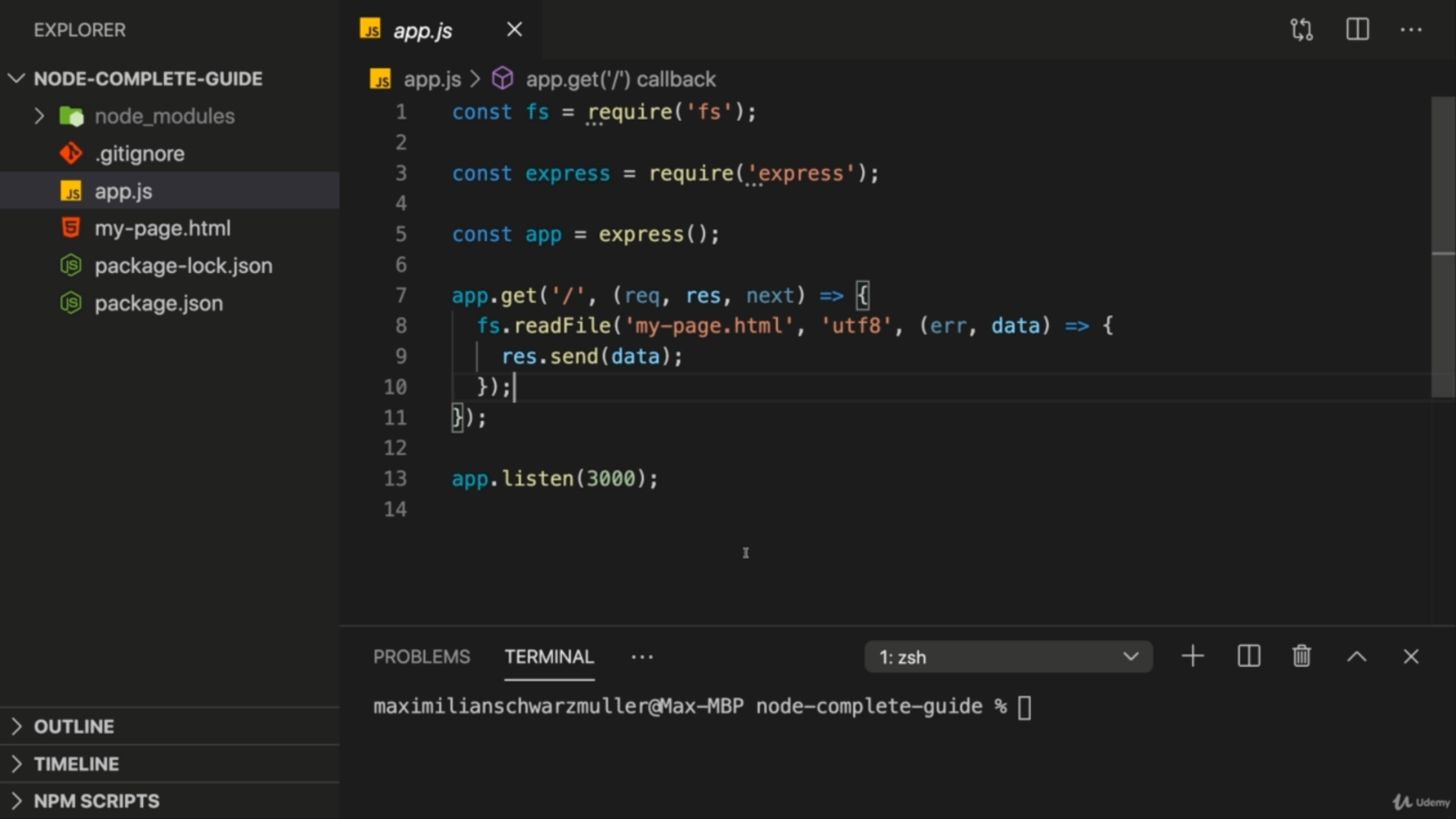
- ES Modules
- Promises in Core APIs

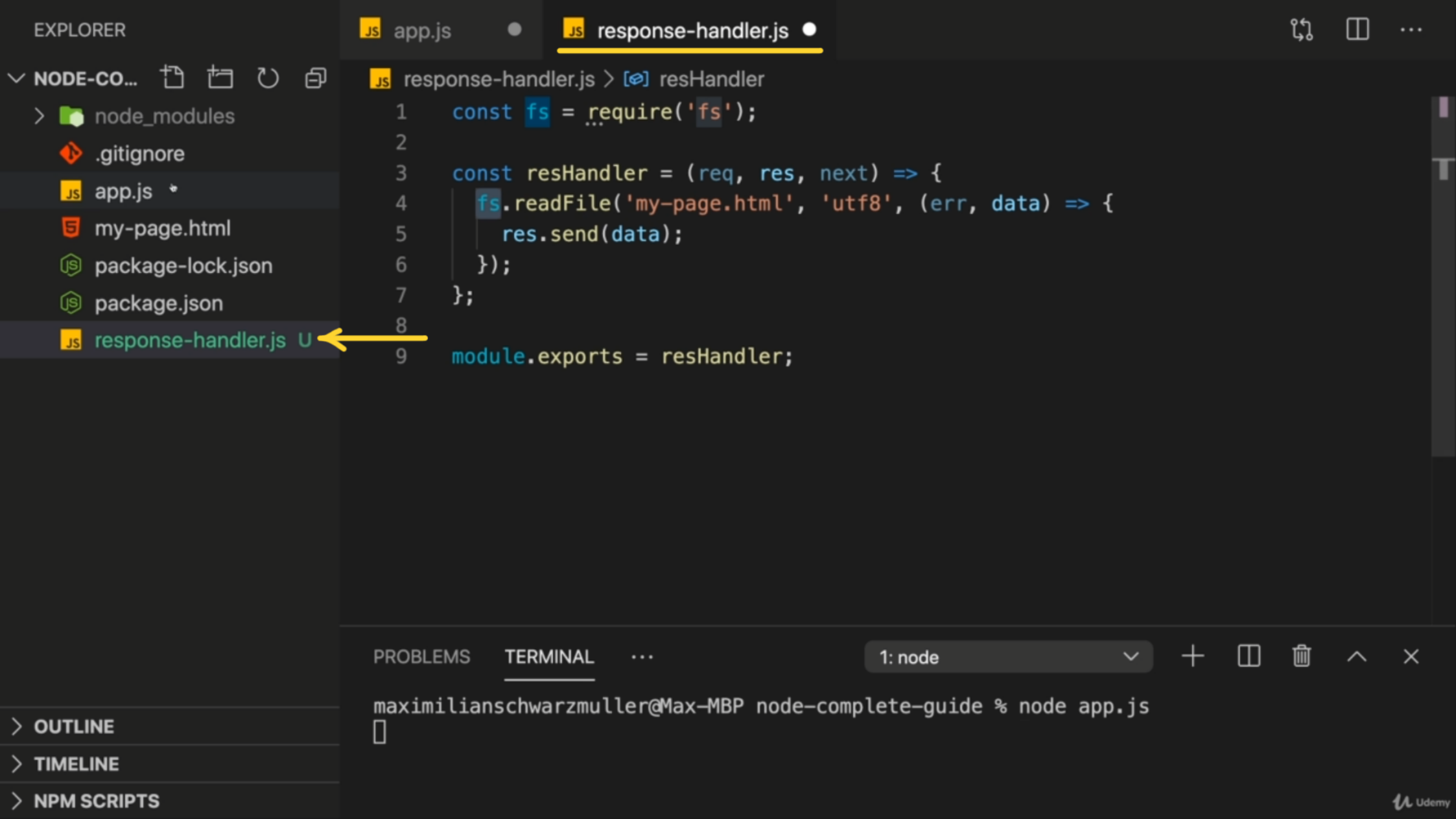
What are "ES Modules"?

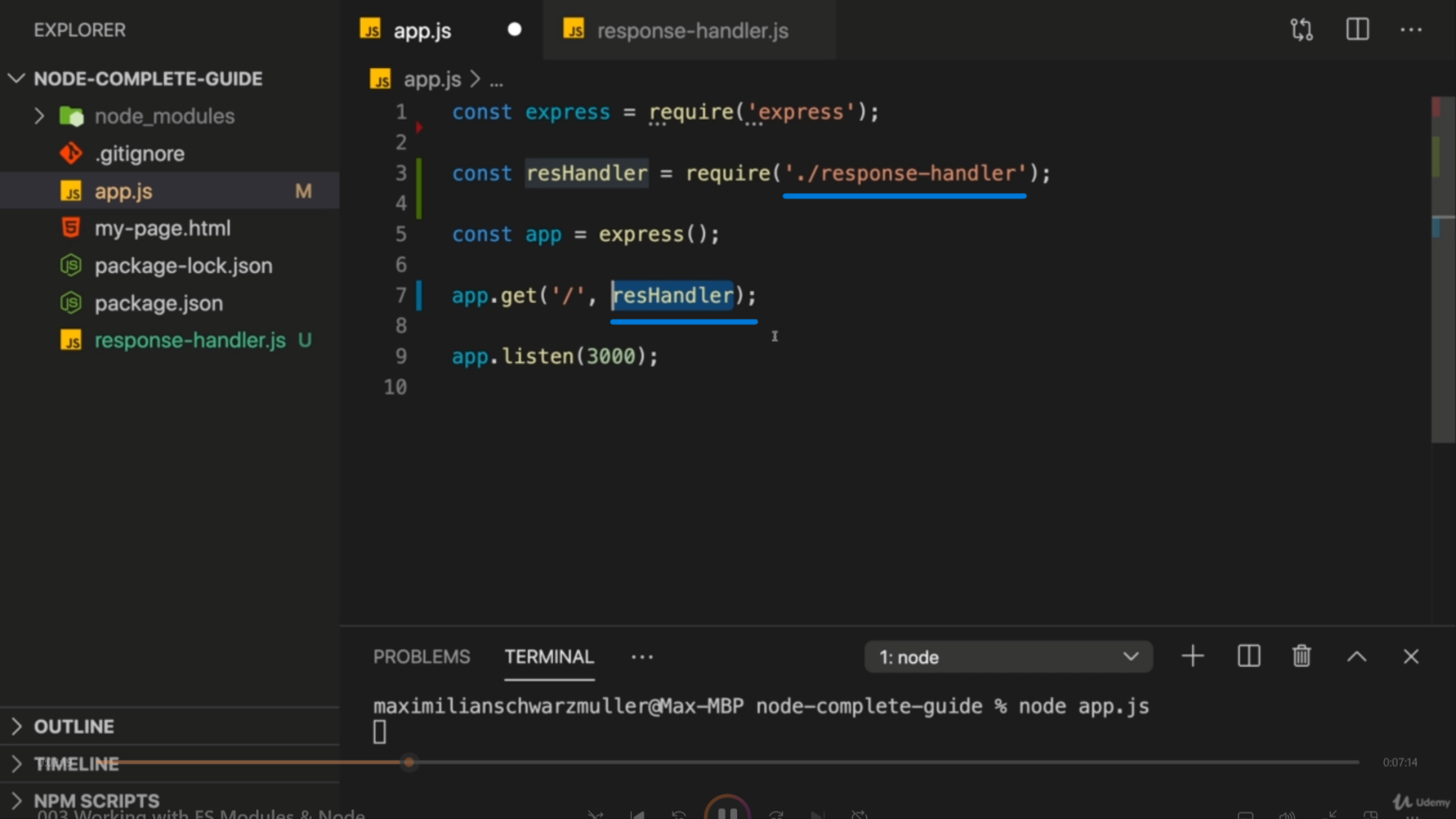
Import/ Export Syntax for modern JavaScript in the Browser

```
export const doSomething = () => { … };
```

```
import { doSomething } from 'my-file';
```


EXPLORER
app.js
NODE-COMPLETE-GUIDE
node_modules
.gitignore
app.js
my-page.html
package-lock.json
package.json
app.js > app.get('/') callback
1 const fs = require('fs');
2
3 const express = require('express');
4
5 const app = express();
6
7 app.get('/', (req, res, next) => {
8 fs.readFile('my-page.html', 'utf8', (err, data) => {
9 res.send(data);
10 });
11 });
12
13 app.listen(3000);
14
PROBLEMS
TERMINAL
1: zsh
maximilianschwarzmuller@Max-MBP node-complete-guide %
OUTLINE
TIMELINE
NPM SCRIPTS


EXPLORER
app.js
response-handler.js
NODE-CO...
node_modules
.gitignore
app.js
my-page.html
package-lock.json
package.json
response-handler.js U
response-handler.js > resHandler
1 const fs = require('fs');
2
3 const resHandler = (req, res, next) => {
4 fs.readFile('my-page.html', 'utf8', (err, data) => {
5 res.send(data);
6 });
7 };
8
9 module.exports = resHandler;
PROBLEMS
TERMINAL
1: node
maximilianschwarzmuller@Max-MBP node-complete-guide % node app.js
OUTLINE
TIMELINE
NPM SCRIPTS


EXPLORER
app.js
response-handler.js
NODE-COMPLETE-GUIDE
node_modules
.gitignore
app.js M
my-page.html
package-lock.json
package.json
response-handler.js U
app.js > ...
1 const express = require('express');
2
3 const resHandler = require('./response-handler');
4
5 const app = express();
6
7 app.get('/', resHandler);
8
9 app.listen(3000);
10
PROBLEMS
TERMINAL
1: node
maximilianschwarzmuller@Max-MBP node-complete-guide % node app.js
OUTLINE
TIMELINE
NPM SCRIPTS
0:07:14

The following example of an ES module imports the function from `addTwo.mjs`:

```
// app.mjs
import { addTwo } from './addTwo.mjs';

// Prints: 6
console.log(addTwo(4));
```

Node.js fully supports ECMAScript modules as they are currently specified and provides limited interoperability between them and the existing module format, CommonJS.

Node.js contains support for ES Modules based upon the Node.js EP for ES Modules and the ECMAScript-modules implementation.

Expect major changes in the implementation including interoperability support, specifier resolution, and default behavior.

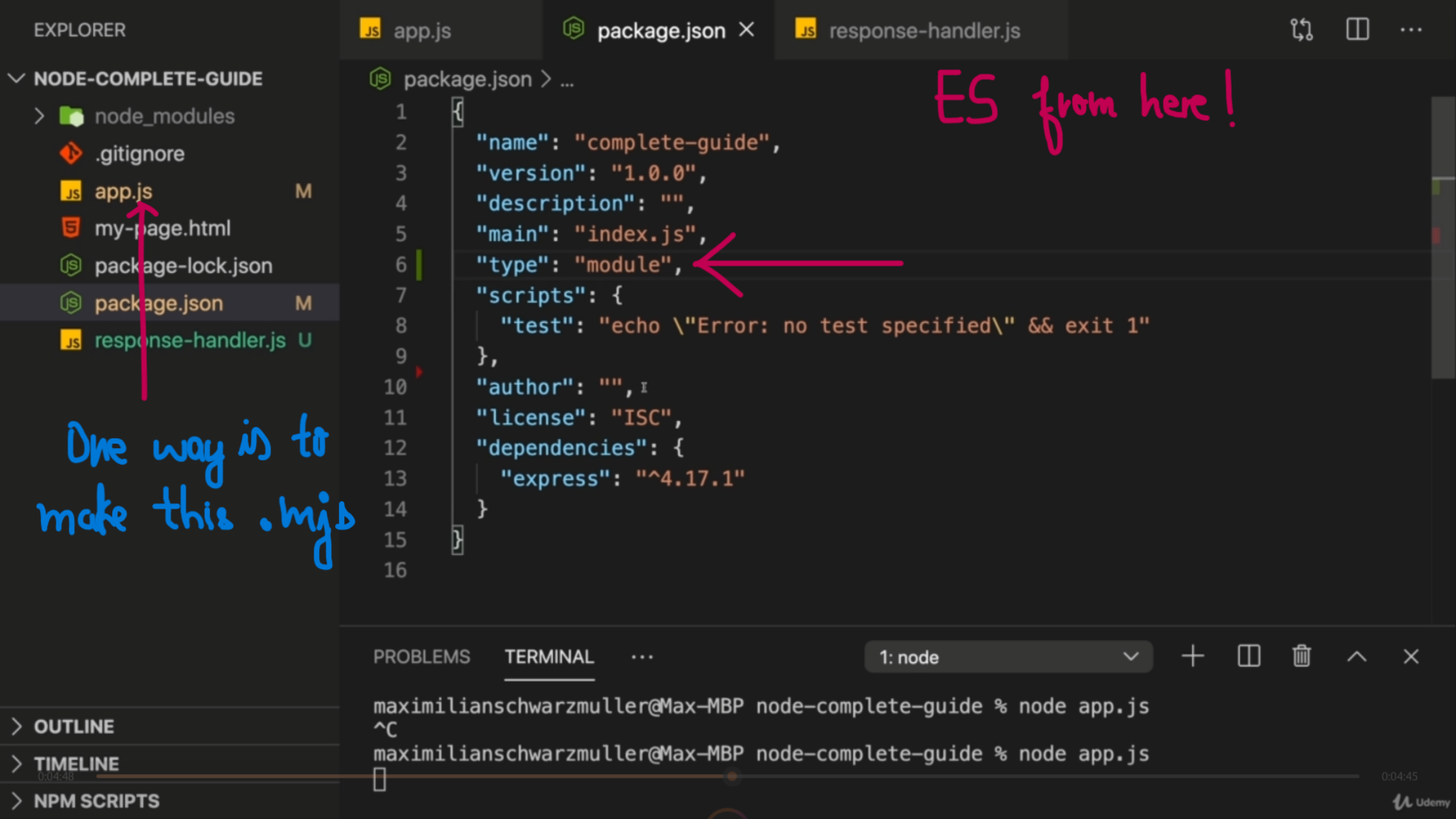
Enabling

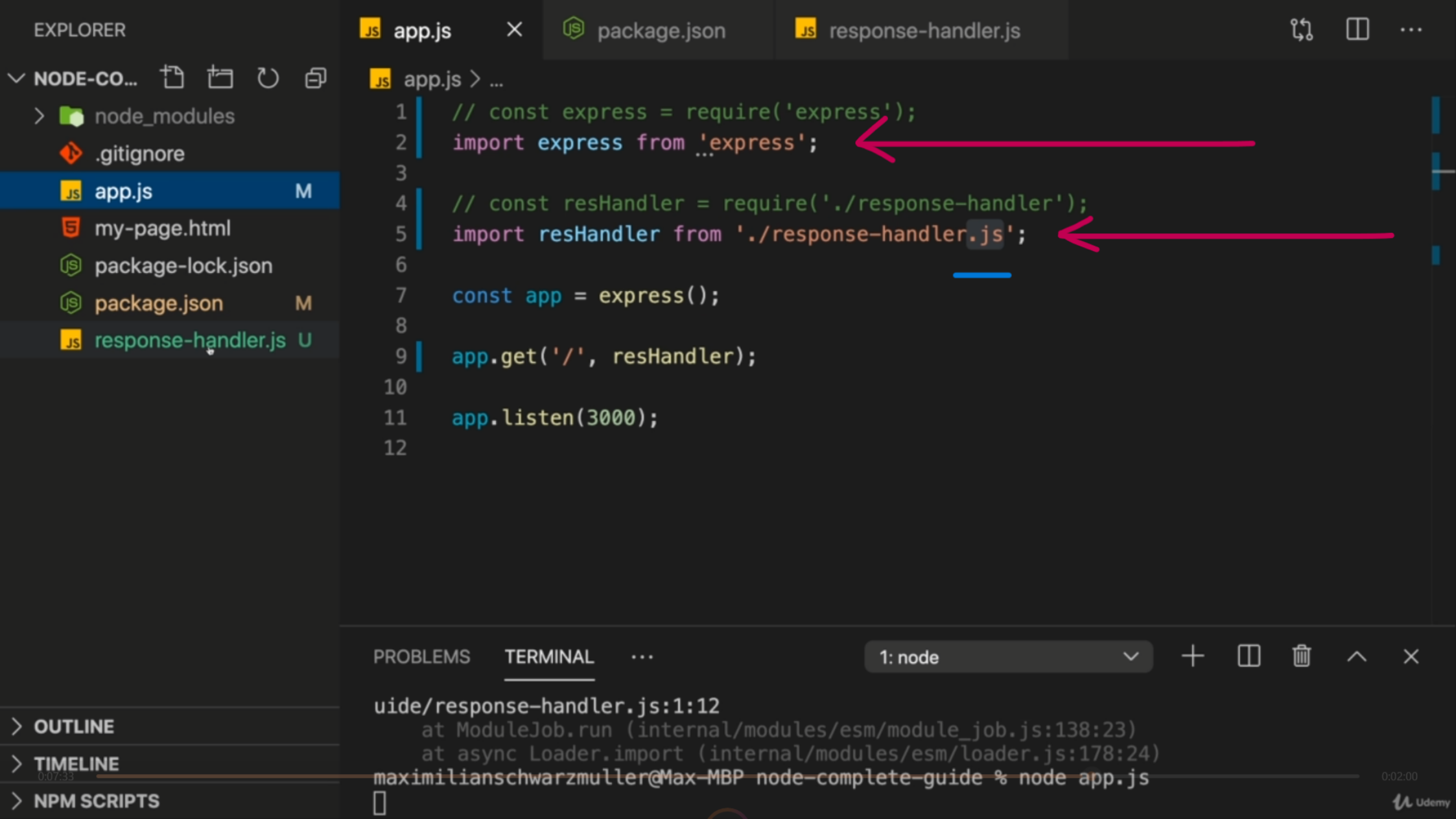
Experimental support for ECMAScript modules is enabled by default. Node.js will treat the following as ES modules when passed to `node` as the initial input, or when referenced by `import` statements within ES module code:

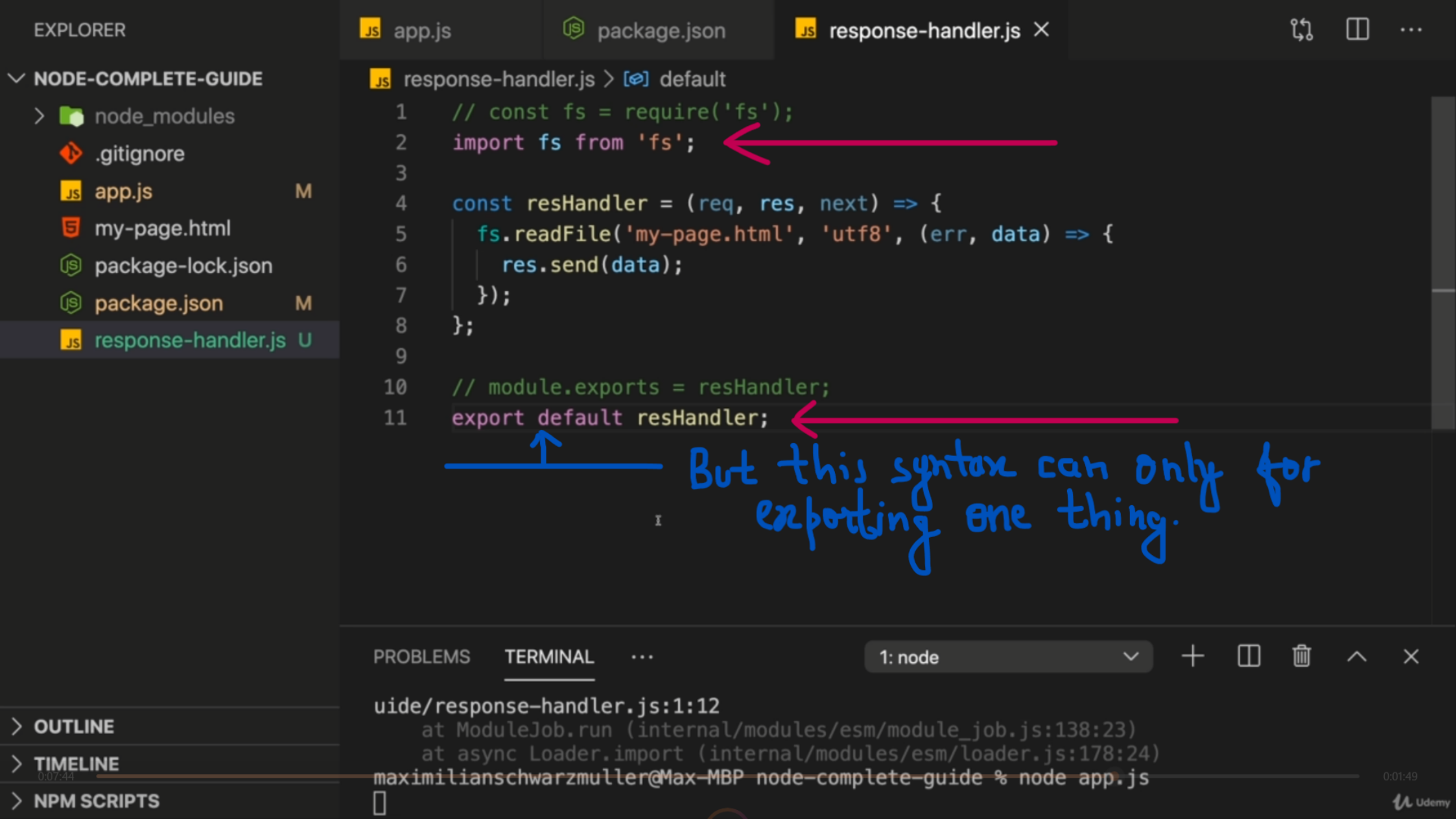
- Files ending in `.mjs`.
- Files ending in `.js` when the nearest parent `package.json` file contains a top-level field `"type"` with a value of `"module"`.
- Strings passed in as an argument to `--eval`, or piped to `node` via `STDIN`, with the flag `--input-type=module`.

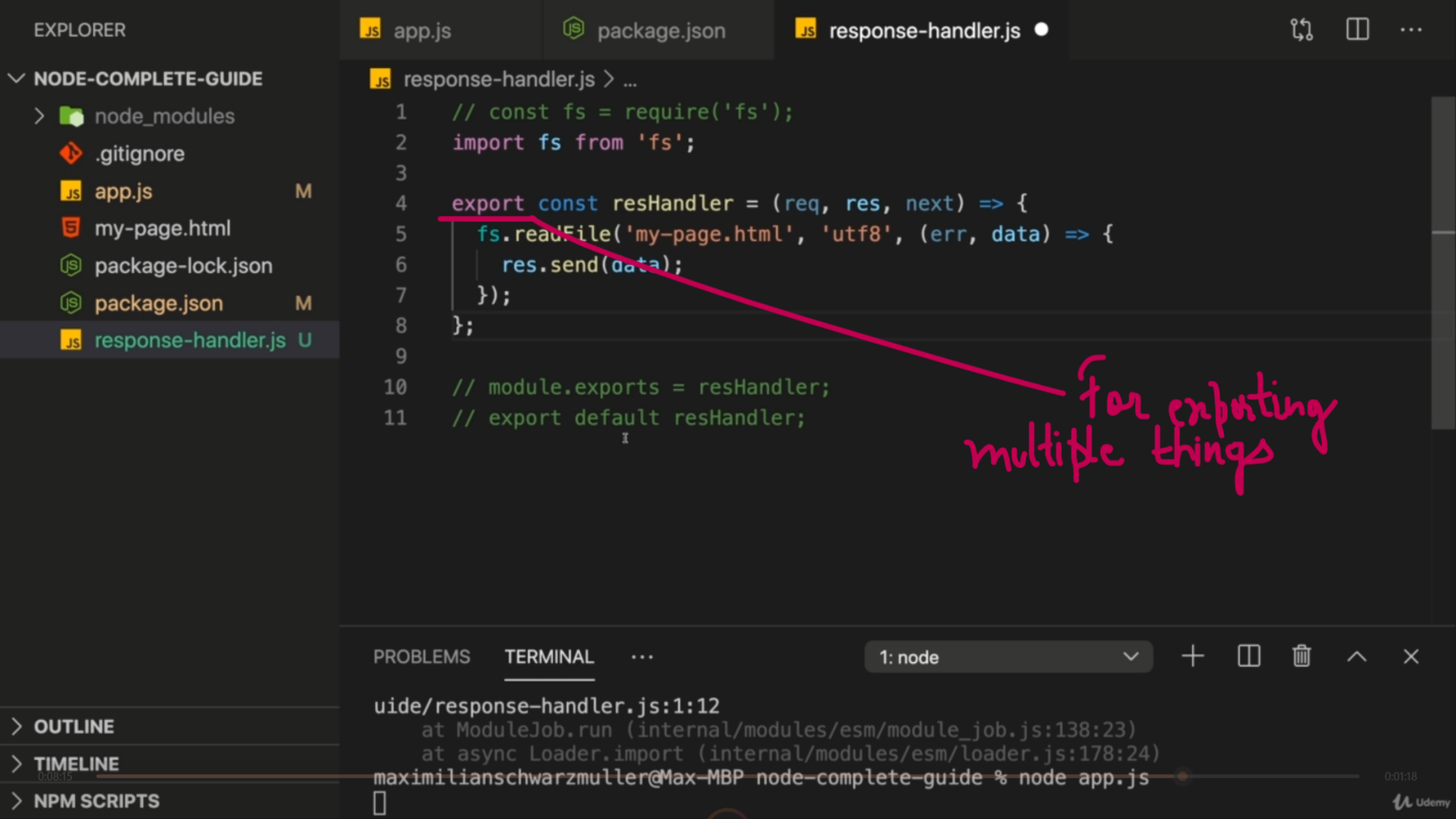
Node.js will treat as CommonJS all other forms of input, such as `.js` files where the nearest parent `package.json` file contains no top-level `"type"` field, or string input without the flag `--input-type`. This behavior is to preserve backward compatibility. However, now that Node.js supports both CommonJS and ES modules, it is best to be explicit whenever possible. Node.js will treat the following as CommonJS when passed to `node` as the initial input, or when referenced by `import` statements within ES module code:

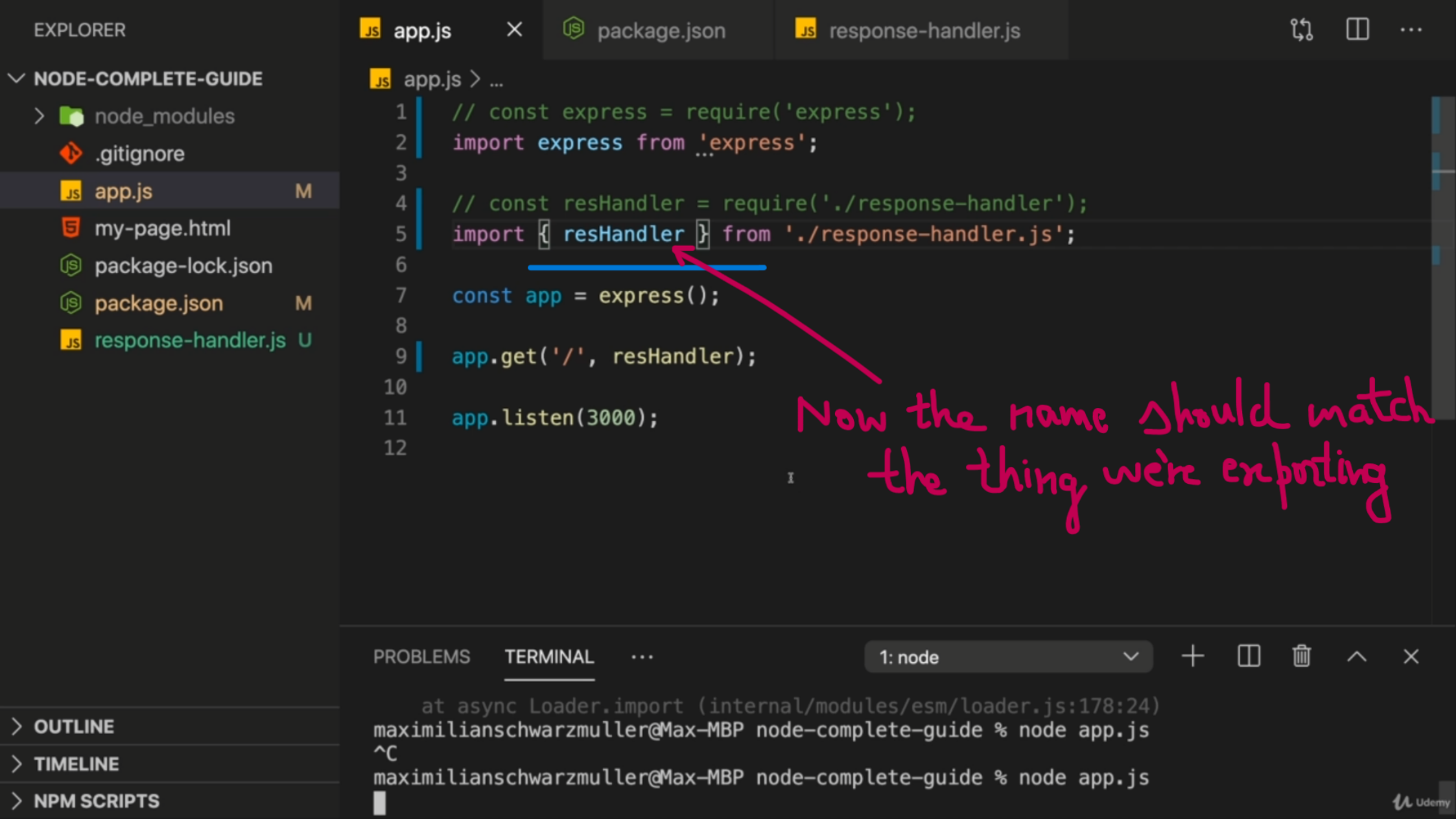
- Files ending in `.cjs`.


EXPLORER
app.js
package.json
response-handler.js
NODE-COMPLETE-GUIDE
node_modules
.gitignore
app.js M
my-page.html
package-lock.json
package.json M
response-handler.js U
package.json > ...
ES from here!
{
"name": "complete-guide",
"version": "1.0.0",
"description": "",
"main": "index.js",
"type": "module",
"scripts": {
"test": "echo \"Error: no test specified\" && exit 1"
},
"author": "",
"license": "ISC",
"dependencies": {
"express": "^4.17.1"
}
}
One way is to make this .mjs
PROBLEMS
TERMINAL
1: node
maximilianschwarzmuller@Max-MBP node-complete-guide % node app.js
^C
maximilianschwarzmuller@Max-MBP node-complete-guide % node app.js
OUTLINE
TIMELINE
NPM SCRIPTS
0:04:48
0:04:45
Udemy


EXPLORER
app.js
package.json
response-handler.js
NODE-CO...
node_modules
.gitignore
app.js M
my-page.html
package-lock.json
package.json M
response-handler.js U
app.js > ...
1 // const express = require('express');
2 import express from 'express';
3
4 // const resHandler = require('./response-handler');
5 import resHandler from './response-handler.js';
6
7 const app = express();
8
9 app.get('/', resHandler);
10
11 app.listen(3000);
12
PROBLEMS
TERMINAL
1: node
uide/response-handler.js:1:12
at ModuleJob.run (internal/modules/esm/module_job.js:138:23)
at async Loader.import (internal/modules/esm/loader.js:178:24)
maximilianschwarzmuller@Max-MBP node-complete-guide % node app.js
OUTLINE
TIMELINE
NPM SCRIPTS


EXPLORER
app.js
package.json
response-handler.js
NODE-COMPLETE-GUIDE
node_modules
.gitignore
app.js M
my-page.html
package-lock.json
package.json M
response-handler.js U
response-handler.js > default
// const fs = require('fs');
import fs from 'fs';
const resHandler = (req, res, next) => {
fs.readFile('my-page.html', 'utf8', (err, data) => {
res.send(data);
});
};
// module.exports = resHandler;
export default resHandler;
But this syntax can only for exporting one thing.
PROBLEMS
TERMINAL
1: node
uide/response-handler.js:1:12
at ModuleJob.run (internal/modules/esm/module_job.js:138:23)
at async Loader.import (internal/modules/esm/loader.js:178:24)
maximilianschwarzmuller@Max-MBP node-complete-guide % node app.js
OUTLINE
TIMELINE
NPM SCRIPTS


EXPLORER
app.js
package.json
response-handler.js
NODE-COMPLETE-GUIDE
node_modules
.gitignore
app.js M
my-page.html
package-lock.json
package.json M
response-handler.js U
response-handler.js > ...
1 // const fs = require('fs');
2 import fs from 'fs';
3
4 export const resHandler = (req, res, next) => {
5 fs.readFile('my-page.html', 'utf8', (err, data) => {
6 res.send(data);
7 });
8 };
9
10 // module.exports = resHandler;
11 // export default resHandler;
for exporting multiple things
PROBLEMS
TERMINAL
1: node
uide/response-handler.js:1:12
at ModuleJob.run (internal/modules/esm/module_job.js:138:23)
at async Loader.import (internal/modules/esm/loader.js:178:24)
maximilianschwarzmuller@Max-MBP node-complete-guide % node app.js
OUTLINE
TIMELINE
NPM SCRIPTS
0:08:15
0:01:18
Udemy


EXPLORER
app.js
package.json
response-handler.js
NODE-COMPLETE-GUIDE
node_modules
.gitignore
app.js M
my-page.html
package-lock.json
package.json M
response-handler.js U
app.js > ...
1 // const express = require('express');
2 import express from 'express';
3
4 // const resHandler = require('./response-handler');
5 import { resHandler } from './response-handler.js';
6
7 const app = express();
8
9 app.get('/', resHandler);
10
11 app.listen(3000);
12
Now the name should match the thing we're exporting
PROBLEMS
TERMINAL
1: node
at async Loader.import (internal/modules/esm/loader.js:178:24)
maximilianschwarzmuller@Max-MBP node-complete-guide % node app.js
^C
maximilianschwarzmuller@Max-MBP node-complete-guide % node app.js
OUTLINE
TIMELINE
NPM SCRIPTS

Mandatory file extensions

A file extension must be provided when using the `import` keyword. Directory indexes (e.g. `'./startup/index.js'`) must also be fully specified.

This behavior matches how `import` behaves in browser environments, assuming a typically configured server.

No `NODE_PATH`

`NODE_PATH` is not part of resolving `import` specifiers. Please use symlinks if this behavior is desired.

No `require`, `exports`, `module.exports`, `__filename`, `__dirname`

These CommonJS variables are not available in ES modules.

`require` can be imported into an ES module using `module.createRequire()`.

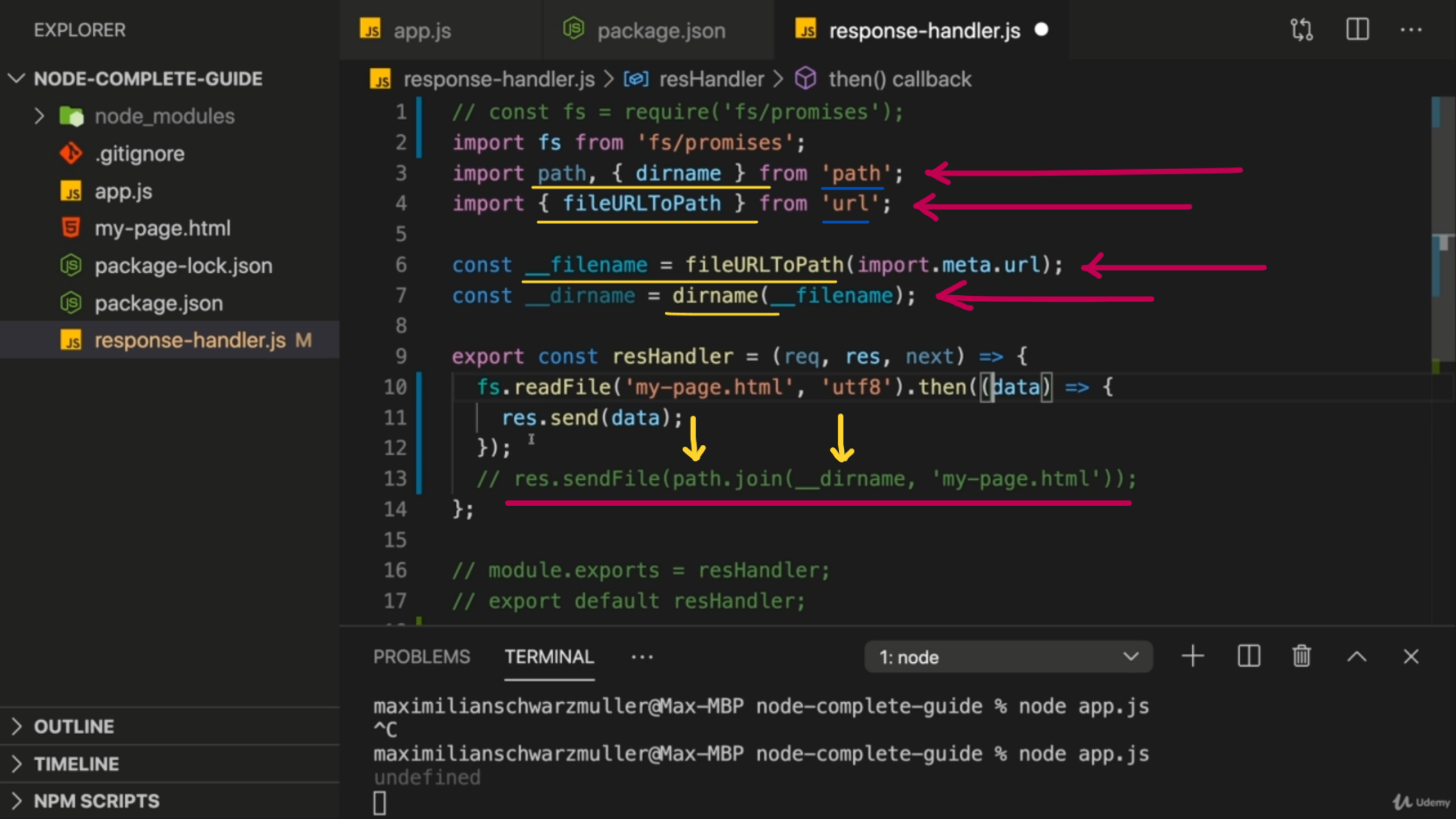
Equivalents of `__filename` and `__dirname` can be created inside of each file via `import.meta.url`.

```
import { fileURLToPath } from 'url';
import { dirname } from 'path';

const __filename = fileURLToPath(import.meta.url);
const __dirname = dirname(__filename);
```

No `require.resolve`

Former use cases relying on `require.resolve` to determine the resolved path of a module can be supported via `import.meta.resolve`, which is experimental and supported via the `--experimental-import-meta-resolve` flag:


EXPLORER
app.js
package.json
response-handler.js
NODE-COMPLETE-GUIDE
node_modules
.gitignore
app.js
my-page.html
package-lock.json
package.json
response-handler.js M
response-handler.js > resHandler > then() callback
// const fs = require('fs/promises');
import fs from 'fs/promises';
import path, { dirname } from 'path';
import { fileURLToPath } from 'url';
const __filename = fileURLToPath(import.meta.url);
const __dirname = dirname(__filename);
export const resHandler = (req, res, next) => {
fs.readFile('my-page.html', 'utf8').then((data) => {
res.send(data);
});
// res.sendFile(path.join(__dirname, 'my-page.html'));
};
// module.exports = resHandler;
// export default resHandler;
PROBLEMS
TERMINAL
1: node
maximilianschwarzmuller@Max-MBP node-complete-guide % node app.js
^C
maximilianschwarzmuller@Max-MBP node-complete-guide % node app.js
undefined
OUTLINE
TIMELINE
NPM SCRIPTS

EXPLORER

app.js | package.json | response-handler.js ●

NODE-COMPLETE-GUIDE

- node_modules
- .gitignore
- app.js
- my-page.html
- package-lock.json
- package.json
- response-handler.js M

response-handler.js > resHandler

```
1  // const fs = require('fs/promises');
2  import fs from 'fs/promises';
3  import path, { dirname } from 'path';
4  import { fileURLToPath } from 'url';
5
6  const __filename = fileURLToPath(import.meta.url);
7  const __dirname = dirname(__filename);
8
9  export const resHandler = (req, res, next) => {
10   fs.readFile('my-page.html', 'utf8')
11     .then((data) => {
12       res.send(data);
13     })
14     .catch((err) => {
15       console.log(err);
16     });
17   // res.sendFile(path.join(__dirname, 'my-page.html'));
```

PROBLEMS TERMINAL ··· 1: node

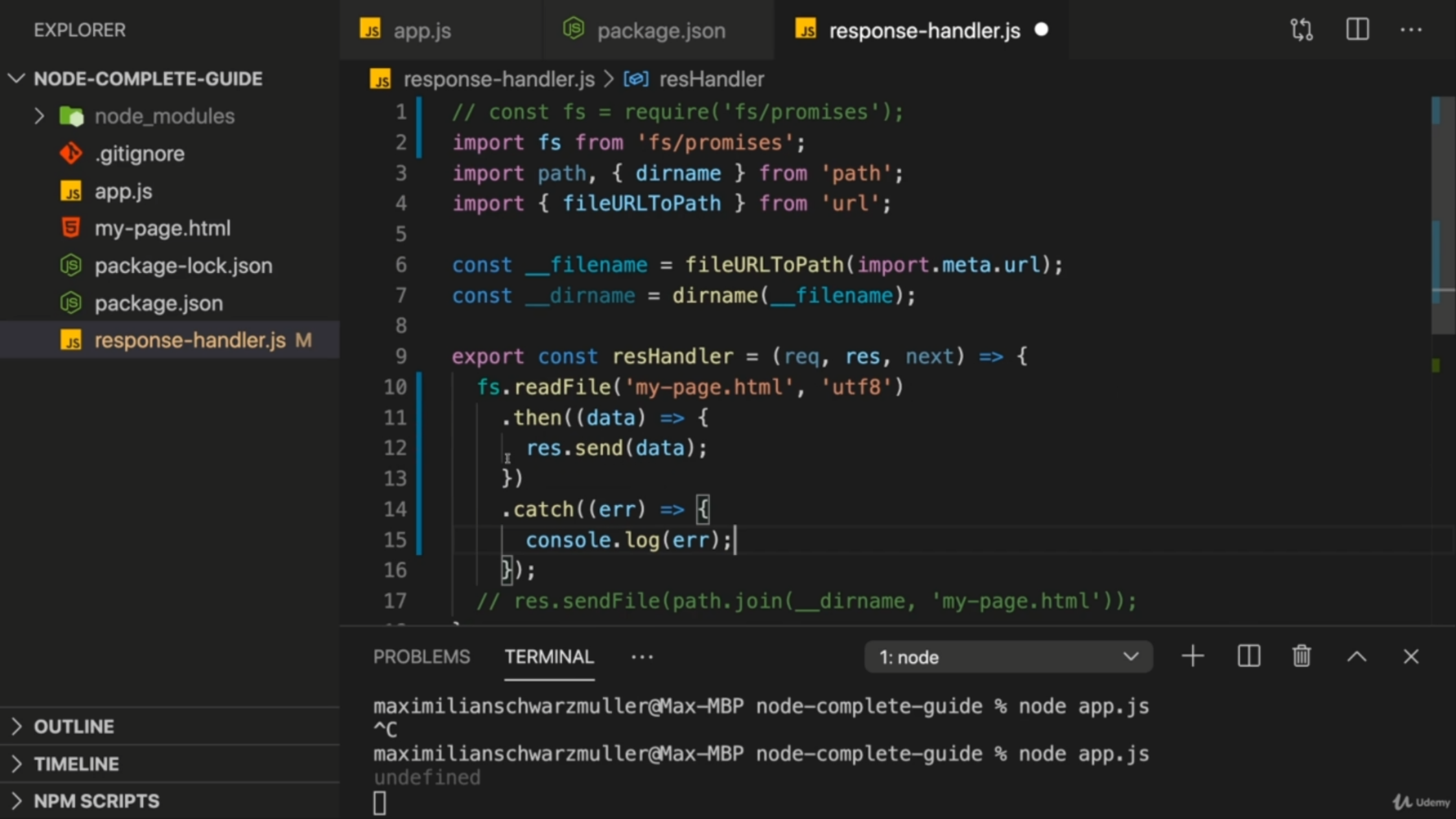
```
maximilianschwarzmuller@Max-MBP node-complete-guide % node app.js
^C
maximilianschwarzmuller@Max-MBP node-complete-guide % node app.js
undefined
```

OUTLINE

TIMELINE

NPM SCRIPTS

Udemy


EXPLORER
app.js
package.json
response-handler.js
NODE-COMPLETE-GUIDE
node_modules
.gitignore
app.js
my-page.html
package-lock.json
package.json
response-handler.js M
response-handler.js > resHandler
// const fs = require('fs/promises');
import fs from 'fs/promises';
import path, { dirname } from 'path';
import { fileURLToPath } from 'url';
const __filename = fileURLToPath(import.meta.url);
const __dirname = dirname(__filename);
export const resHandler = (req, res, next) => {
fs.readFile('my-page.html', 'utf8')
.then((data) => {
res.send(data);
})
.catch((err) => {
console.log(err);
});
// res.sendFile(path.join(__dirname, 'my-page.html'));
PROBLEMS
TERMINAL
1: node
maximilianschwarzmuller@Max-MBP node-complete-guide % node app.js
^C
maximilianschwarzmuller@Max-MBP node-complete-guide % node app.js
undefined
OUTLINE
TIMELINE
NPM SCRIPTS