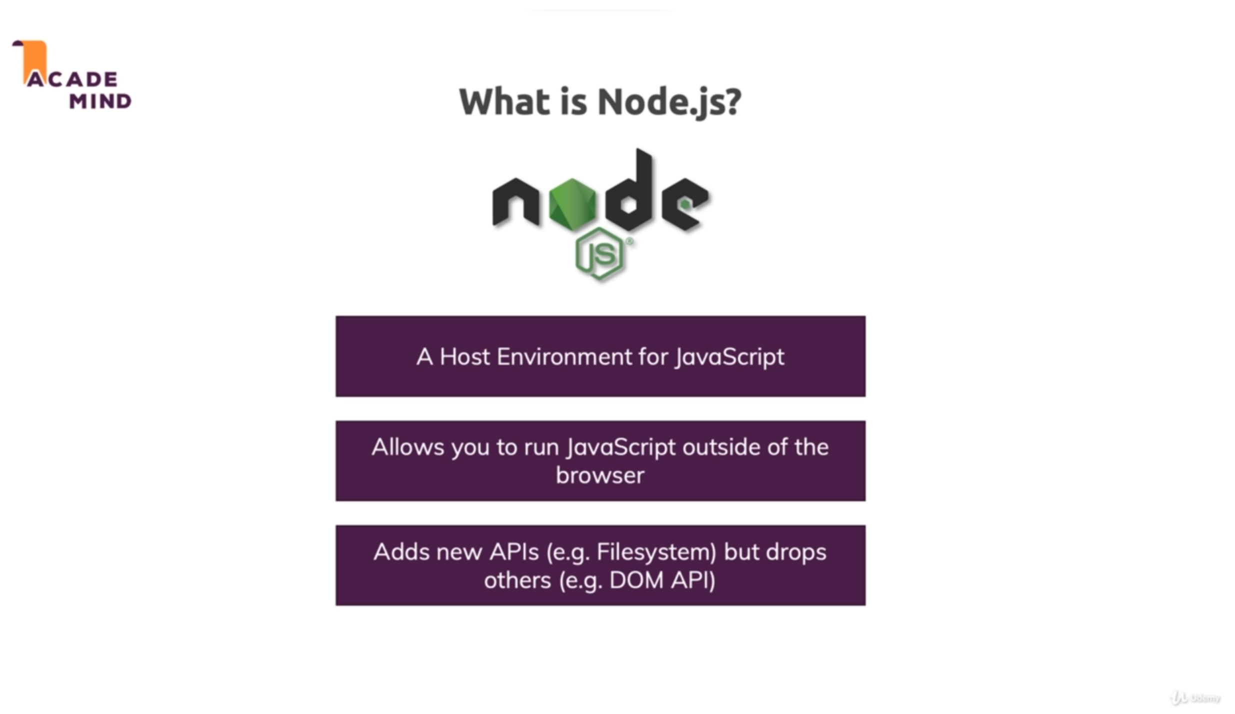
Node.js & Express.js Refresher

By - Shashwat Agrawal

Introduction



What is Node.js?

A Host Environment for JavaScript

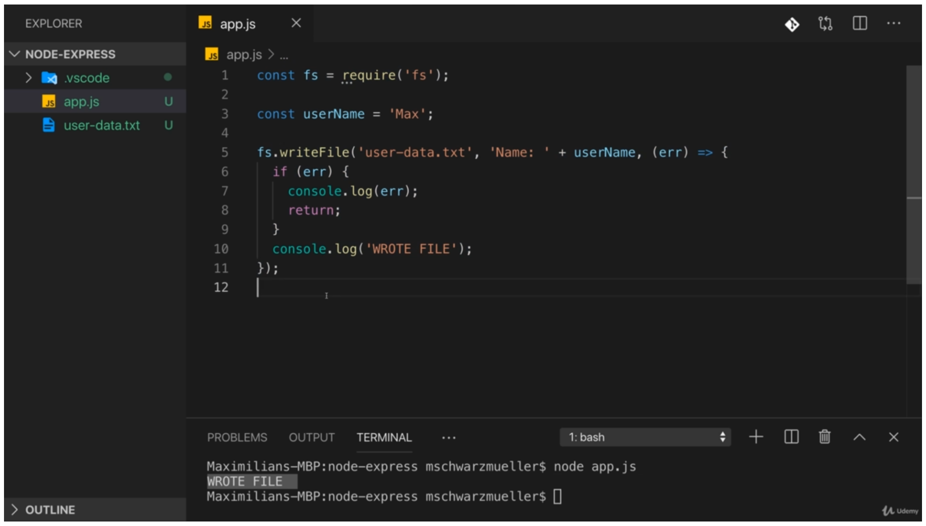
Allows you to run JavaScript outside of the browser

Adds new APIs (e.g. Filesystem) but drops others (e.g. DOM API)

- Node.js basically creates an environment to execute javascript on server machines (or local machines).
- One popular use case of node.js is to build web servers that handle incoming requests, talk to databases, produce responses, etc.
- Various APIs do not exist in Node.js but exist in JS run through browsers. (Eg. the alert function works in the browser but does not work in the node.js environment).

Node.js

- A simple example of using JavaScript in Node.js



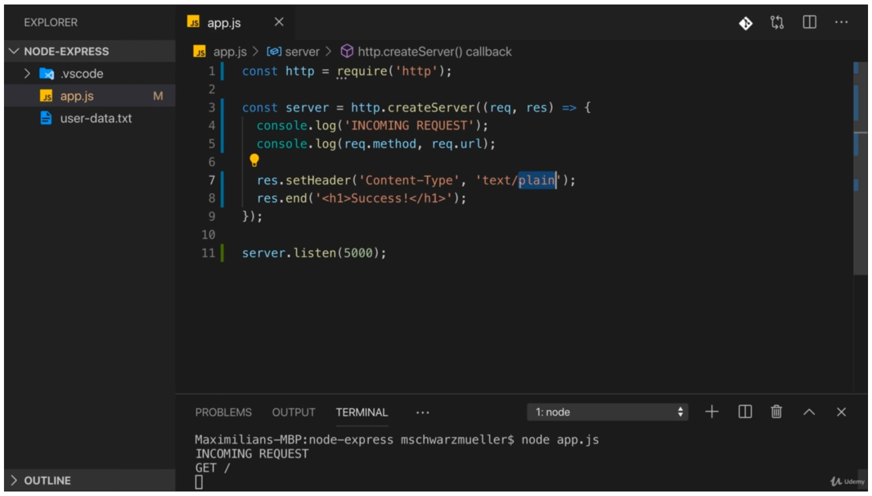
```javascript
const fs = require('fs');

const userName = 'Max';

fs.writeFile('user-data.txt', 'Name: ' + userName, (err) => {
  if (err) {
    console.log(err);
    return;
  }
  console.log('WROTE FILE');
});
```

```
Maximilians-MBP:node-express mschwarzmueller$ node app.js
WROTE FILE
Maximilians-MBP:node-express mschwarzmueller$
```

- ‘fs’ is a module (collection of files) provided by node.js
- We imported that module to use its functionality, see through the documentation of the file system module in node.js documentation and got to know the parameters to be passed here.
- A new file has been made with the text passed to it and a function is made to log errors if any.
- There are many such modules present in the node for example, creating a new file, writing in a file, clear all files in the folder, audit files, etc.

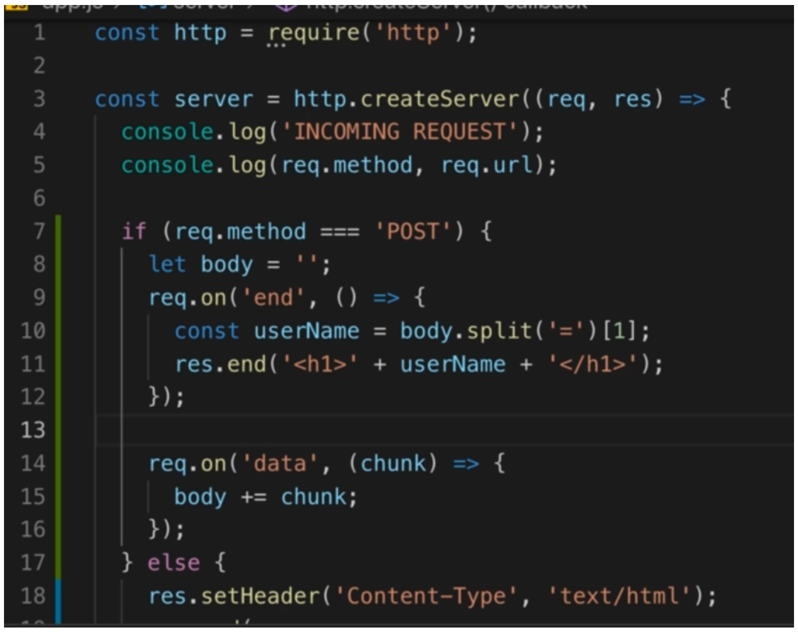
Sending Requests & Responses

- We actually don’t want to deal with vanilla JavaScript as sending requests and getting responses using node.js is a huge task to be carried on.
- If you want, learn Node.js separately.
- An example of just sending one request and getting a response takes too many lines of code.



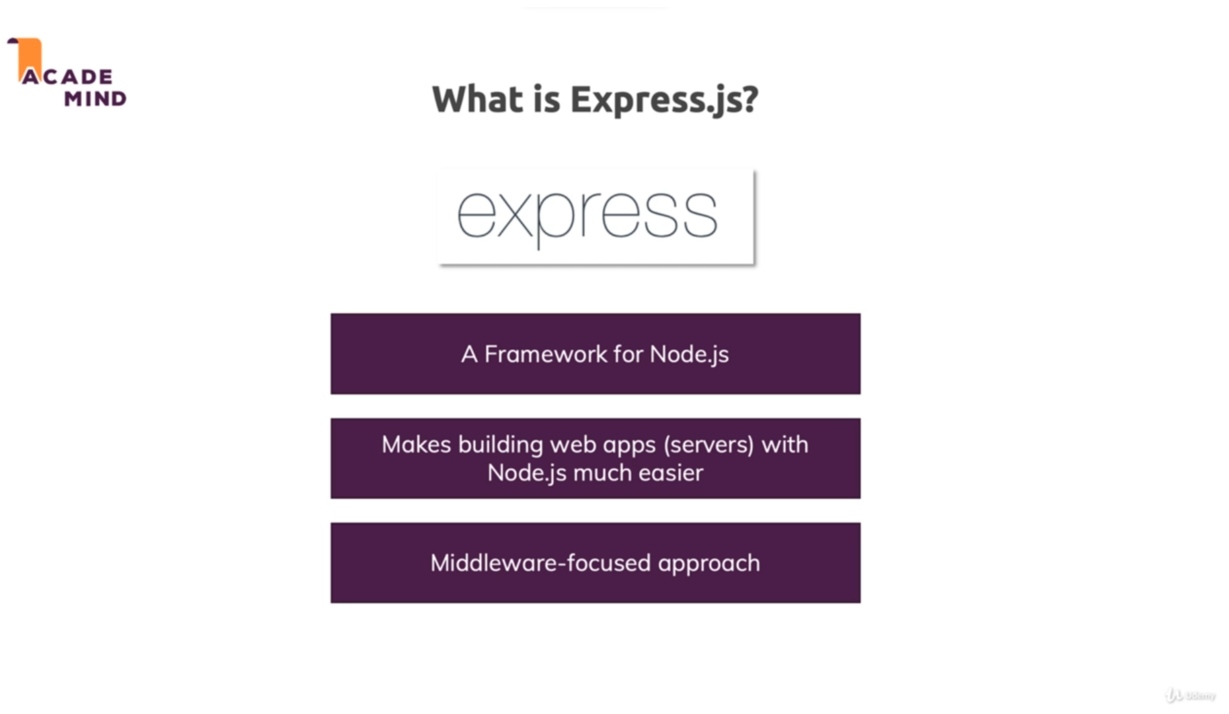
```javascript
const http = require('http');

const server = http.createServer((req, res) => {
  console.log('INCOMING REQUEST');
  console.log(req.method, req.url);

  res.setHeader('Content-Type', 'text/plain');
  res.end('<h1>Success!</h1>');
});

server.listen(5000);
```

```
Maximilians-MBP:node-express mschwarzmueller$ node app.js
INCOMING REQUEST
GET /
```



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

const server = http.createServer((req, res) => {
  console.log('INCOMING REQUEST');
  console.log(req.method, req.url);

  if (req.method === 'POST') {
    let body = '';
    req.on('end', () => {
      const userName = body.split('=')[1];
      res.end('<h1>' + userName + '</h1>');
    });

    req.on('data', (chunk) => {
      body += chunk;
    });
  } else {
    res.setHeader('Content-Type', 'text/html');
```

Express.js



What is Express.js?

A Framework for Node.js

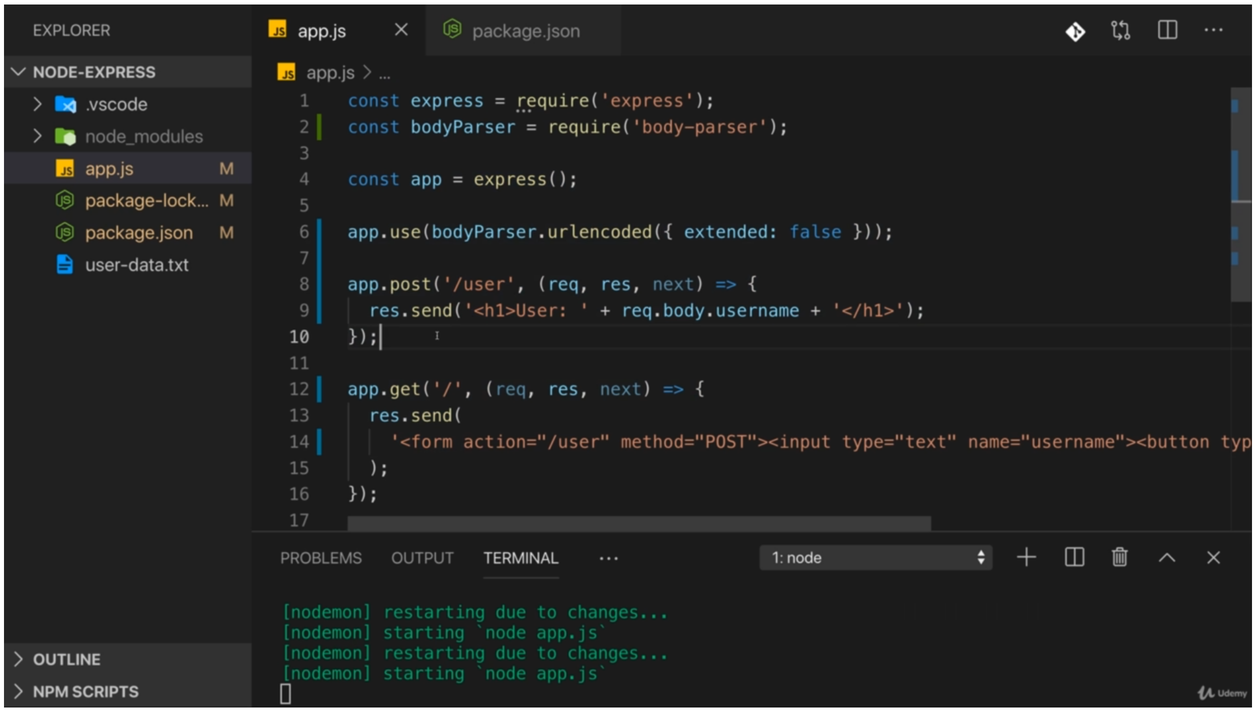
Makes building web apps (servers) with Node.js much easier

Middleware-focused approach

- npm is a tool that is installed together with node.js and is a node package manager which holds and helps in managing node modules.

- It is necessary to initialize npm to add express and use its functionalities. Basically, npm is used to manage third-party libraries like express, that's why we use it for react also.
- An object is a collection of functions and data. A function is a collection of instructions and commands.

- Every middleware is either meant to perform something or send some response. next() function is called when we want to forward the request to another middleware.
- Example of express middleware:



```
const express = require('express');
const bodyParser = require('body-parser');

const app = express();

app.use(bodyParser.urlencoded({ extended: false }));

app.post('/user', (req, res, next) => {
  res.send('<h1>User: ' + req.body.username + '</h1>');
});

app.get('/', (req, res, next) => {
  res.send(
    '<form action="/user" method="POST"><input type="text" name="username"><button typ
  );
});
```

```
[nodemon] restarting due to changes...
[nodemon] starting `node app.js`
[nodemon] restarting due to changes...
[nodemon] starting `node app.js`
```

middleware functions are executed when the routes are called but not before that.