Node.js

- An environment to run JS outside browser
- Created in 2009
- Built on chrome's V8 JS Engine
- Big community
- Full stack

V8 - Every browser needs a tool that compiles our code down to machine code and chrome uses V8 as this tool.

- server side.

Browser	NodeJS
DOM	No Dom
Window	No Window
Interactive Apps, GUI	Server side app, no GUI
No filesystem	Can access filesystem
Fragmentation	Versions
ES6 Modules	CommonJS (by default)

How to start

- on terminal type 'node'

Repl → Read Eval Print loop & CLI executable

How to run →

node filename.js or node filename

eg: node app.js

app.js.

```
const amount = 9
if(amount<10)
 {
   console.log('Small number')
 }
else
 {
  console.log('Big number')
 }
console.log('hey, Its my first node app')
```

Global variables — NO WINDOW !!

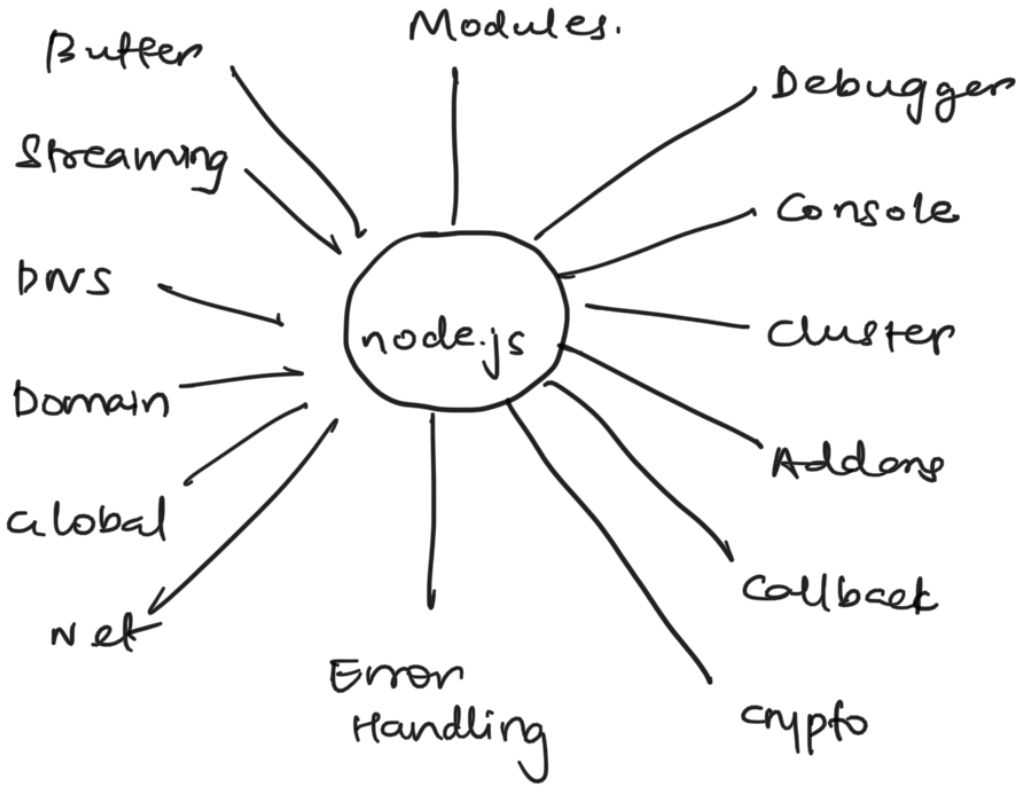
- can access anywhere in code

* __dirname - path to current directory
* __filename - file name
* require - function to use modules. (CommonJS)

* module - info about current module (file)
* process - info about env where the program is being executed.

```
setInterval(() => {
    console.log('Hello woorld')
3,1000)
```
→ will output helloworld after every 1000ms

Some important parts of node.js.



Features

1. Extremely fast code execution
2. I/O is Asynchronous and Event driven
3. Single threaded
4. Highly Scalable.
5. No buffering
6. Open source
7. License. - released under MIT License

NodeJS Console

console.log() → display simple message on console
console.error() → render error message on console
console.warn() → display warning message on console

Module in node:

modularize the code, separate into files

* Module: Encapsulated code (share only minimum)
* every file is a module by default.

names.js.

```
// local
const secret = "SUPER SECRET"
// share
const john = 'john'
```

```
const peter = 'peter'
```

it is an array in modules, will have shared values in it.

```
module.exports = {john, peter}
```

key - john, value: 'john'

key val = 'peter'

utils.js

```
const sayHi = (name) => {
    console.log (`Hello ${name}`)
}

module.exports = sayHi
```

module.js

```
const names = require('./names')
const sayHi = require('./utils')
// const {john, peter} = require('./names')
sayHi ('susan')
sayHi (names.john)
sayHi(names.peter)

const data = require('./alternative-flavor')
console.log(data).          // o/p - {items: ['item1', 'item2'],
                                   singlePerson : {name: 'bob'}}
require('./07-mind-grenade') → even if you dont assign to a variable, the code inside the module will run.
```

alternative-flavor.js

```
module.export.items = ['item1', 'item2']
const person = {
    name : 'bob',
}
module.exports.singlePerson = person.
```

* Remember: when you import a module, you actually invoke it.

Built-in modules

os, PATH, fs, HTTP

OS

```
const os = require('os')
const user = os.userInfo()
console.log(user)

// to get system uptime
os.uptime()
console.log(`Uptime is ${os.uptime()}`)
```

os.type() → os name. unix / darwin / windows-nt
os.release() → os release.
os.totalmem() → total memory of system (bytes)
os.freemem() → free memory (in bytes)
many other methods.

PATH

```
const path = require('path')
console.log(path.sep)   — path separator. /or\.
 O/P -  /
```

join all args together.

```
const filePath = path.join('/content/', 'subfolder', 'test.txt')
console.log(filePath)
//op.    /content/subfolder/test.txt
```

last portion of path.

```
const base = path.basename(filePath)
console.log(base)
//OP    test.txt
```

resolve an abs path

```
const absolute = path.resolve(__dirname, 'content', 'subfolder',
                              'test.txt')
console.log(absolute)
```

O/P /User/smilga/Desktop/tutorial/content/subfolder/test.txt

fs — synchronous / asynchronous.

```
const {readFileSync, writeFileSync} = require('fs')
```

fs-sync.js.

```
const first = readFileSync('./content/first.txt', 'utf8')
const second = ——— u ——————— second.txt', 'utf8')

writeFileSync(
      './content/result.txt',        → file to write into.
      `Here is result: ${first}, ${second}`,   → content
      {flag = 'a'}                   → append.
)
```

Asynchronous. fs-async.js

Callback - we run it when we are done

```
const {readFile, writeFile} = require('fs')
console.log('start')
readFile('./contents/f1.txt', 'utf8', (err, result) => {
  if (err) {
    console.log(err)
```

```
    })
const first = result.
// read second.txt same as above.

writeFile(
    './content/result.txt',
    'Here is result : ${first}, ${seeend}',
    (err, result) => {
        if(err)
        {
            log error
            return
        }
        console.log('done with this task')
    }
)
})
console.log('starting next task')
```

op- start
Starting next task
done with this task.

↑ is a tedious approach, instead use promise & async, await

HTTP

```
const http = require('http')
const server = http.createserver((req, res) => {
    if (req.url === '/')
    {
        res.end('Homepage')
    }
    else if (req.url === '/about')
    {
        res.end('About')
    }
    else{
        res.end('
        <h1> oops!</h1>
        <p> cant find page </p>
        <a href = "/"> back home </a>
        ')
    }
})

server.listen(5000);
```

NPM Node Package Manager

When we install node we automatically install npm

npm enables us to do 3 things -
1) reuse our code in other projects
2) use code written by other developers.
3) share our own soln with other developers.

package → reusable code.
↳ a folder that contains js code
package == module == dependency

npm --version.

// local dependency - use only in this particular project
npm i <packageNames>

// global dependency - use in any project
npm install -g <packageName>.
// sudo npm install -g <packageName> //on mac.

package.json
- manifest file (store important info about proj/package)
3 ways to create
1. manually (create it in root, create properties etc).
2. npm init (step by step, press enter to skip)
3. npm init -y (everything default).

* package.json will have dependencies we add using npm i.

* The dependencies will be in node_modules folder.

When you push your code, you don't need to add the node_modules folder.
When we type npm install all dependencies inside package.json are fetched and node_modules is created.

dev dependency
npm i <packagename> -D
OR npm i <pkgname> --save-dev
eg npm i nodemon -D
→ automatically restarts app on detecting changes.

Why devdependency - we use it only while building our app we dont need them in production.

package.json → scripts.

```
"scripts": {
    "start": "node 01-intro.js",
```
→ npm start will run 01-intro.js.

"dev": "nodemon app.js" -> npm run dev will execute app.js
},

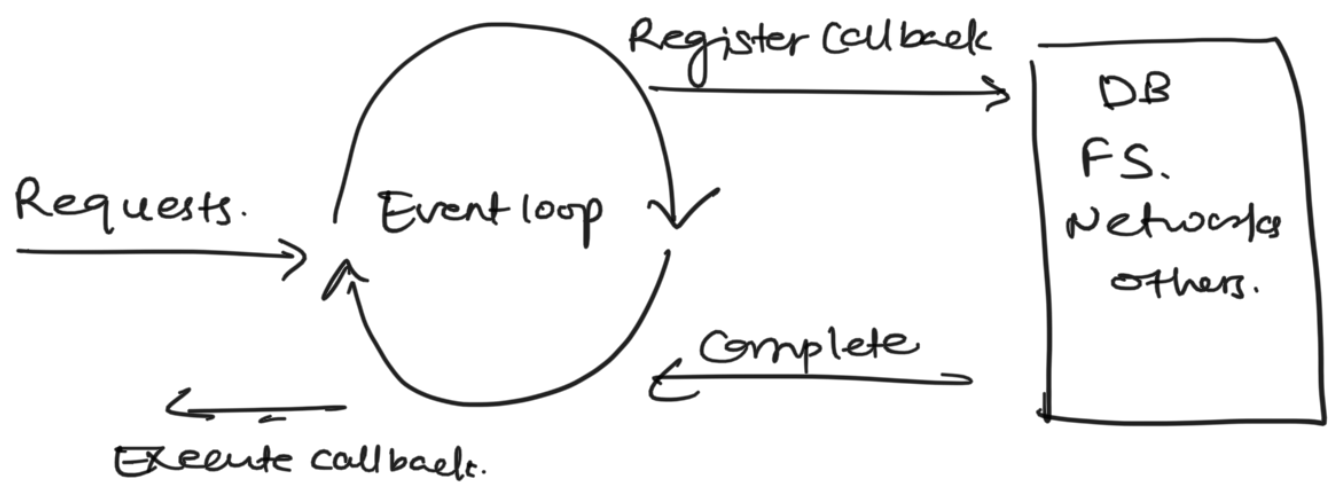
* npm uninstall bootstrap
or you can remove from dependencies in package.json.
delete node_modules and use npm install

npx → execute.
-> a package runner.
npx create-react-app my-app.
cd my-app
npm start

npx preferred over global installation.
npx nodemon app.js.

package_lock.json
So you want users to use same versions of dependencies as you did.
And when you install dependency many modules can be added. eg - bootstrap will add jquery, etc too
So package_lock.json will define versions for all the dependencies (including jquery etc)

Eventloop
Event loop is what allows nodejs to perform non blocking I/O operations - despite the fact that Javascript is single threaded - by offloading operations to system kernel whenever possible.




```
console.log('first')
setTimeout(() => {
    console.log('second')
},0)
console.log('third').
```

o/p - first
third
second

bcoz setTimeOut is asynchronous.

Another eg - readfile → async

eg - setInterval()

```
setInterval (() => {
    console.log ('hello world')
}, 2000)
console.log ('I run first')
```

o/p I run first
Hello world
Hello world
:
∞.

eg.
```
server.listen (5000, () => {
    console.log ('listening on :5000')
})
```

All these codes with async calls turn messy soon.

Alternative

Promise, async, await.

```
const {readfile, writefile} = require('fs').promises
// const util = require ('util')
// const readfilePromise = util.promisify (readfile)
// const writefilePromise = util.promisify (writefile).

const start = async() => {
    try {
        const first = await readfile ('./content/first.txt', 'utf8')
        const second = await readfile ('./content/second.txt', 'utf8')
        await writefile (
            './content/result.txt',
            `This is awesome ${first}, ${second}`,
            {flag :'a'}
        )
        console.log (first second)
    } catch (error) {
        console.log(error)
    }
}
start().
```

use try catch.

OR

```
const getText = (path) => {
    return new Promise ((resolve, reject) => {
        readfile (path, 'utf8', (err, data) => {
```

```
            if(err)
            {
              reject(err)
            }
            else
            {
              resolve(data)
            }
          })
        })
      })
getText('./content/f1.txt')
  .then((result) => console.log(result))
  .catch((err) => console.log(err))
```

not clean use above code instead

Events

Event driven programming → do something on certain event
eg click, hover etc

How is it in server side?
We listen for specific events and register functions that will execute in response to those events.

```
const EventEmitter = require('events');
      ↓class
```

If you want custom event → extend that class
else just create an instance of the class

```
const CustomEmitter = new EventEmitter() // instance

customEmitter.on('response', () => {  → callback fn
  // listen to an event → on
    console.log('data received')
})

customEmitter.emit('response')   // emits on an event with
                                    name 'response'
```

1) We can have multiple on for single event
2) we can also pass parameter to on callback function.
3) The order of on & emit matters.
   we can only emit on listening.

eg.

```
const http = require('http')
const server = http.createServer()
server.on('request', (req, res) => {   → http server will have
    res.end('Welcome')                    this event name.
```

3)

server.listen(5000)

which eventually extends EventEmitter

Streams → used to read or write sequentially.

Extend eventemitter class

Writable → write sequentially

Readable → read —u—

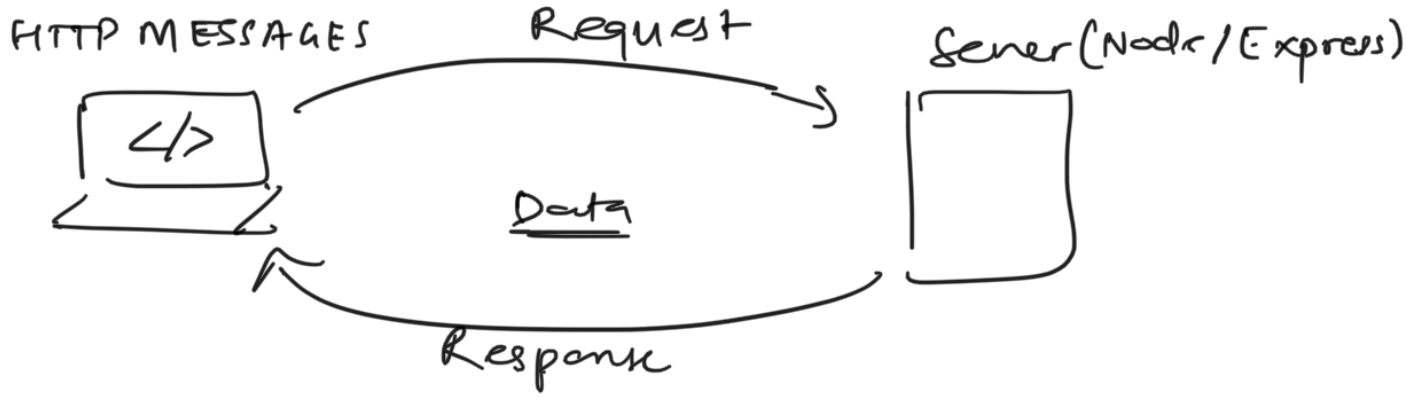
Duplex → read/write —u—

Transform → modify when read/write

If file is too big, use read stream instead of storing files

```
const { createReadStream } = require('fs')
const stream = createReadStream('./content/big.txt')
stream.on('data', (result) => {
        console.log(result)
  3)
stream.on('error', (err) => console.log(err))
```

How web browser work?



Request message

Method — GET /contact HTTP/1.1

optional ← Headers

Body

Response message

Status code

HTTP/1.1 200 OK → Status Text

Headers

Body — optional

Common Port no's.

20 - FTP (Data Transfer)

21 FTP (Command Tool)

22 SSH

25 SMTP

53 DNS

80 HTTP

443 HTTP Secure (HTTPS) HTTP over TLS/SSL

HTTP Response codes.

1. Informational Response 100 - 199

2 Successful Response 200 - 299
3 Redirects 300 - 399
4. Client errors 400 - 499
5 Server errors 500 - 599

MIME - Multipurpose Internet Mail Extensions (MIME Type)
indicates nature & format of a document

Problem with prev http req
1. There is no info on headers.
2. for every url it will give same resp req
/, /contact, /about, /error/info → op → same
or you have to write code for every res in webpage.!!

1→
```
res.writeHead(200, {'content-type': 'text/html'})
res.write('<h1>Hello Home</h1>')
res.end().
```

2→ we will use express.

Express - a minimal and flexible node.js app framework designed to make developing websites, webapps and apis much faster and easier
-It is not a builtin module like http, fs etc

```
npm install express
```
Methods
app.get
post
put
delete
all
use // middleware
listen

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

app.get('/', (req, res) => {
    console.log('hit')
    res.status(200).send('Home Page')
})
similarly /about
app.all('*', (req, res) => {
    res.status(400).send('error')
})
```

```
app.listen(5000, () => {
        console.log('Port 5000')
})
```

```
const path = require('path')
app.use(express.static('./public'))

app.get('/',(req,res) => {
    res.sendfile(__dirname, './nav-app/index.html')
})
```

Express - API vs Server Side Rendering (SSR)

API - JSON	SSR - Template
Send Data	Send Template
Res.JSON()	Res.render()

API - Server provides a
res.json().

```
app.get('/', (req,res) => {
    res.json(`[{name":"Harry"}, {"name":"Susan"}]`)
})
```

JS 05-all-static.js | JS 06-basic-json.js | JS *07-params-query.js* × | JS data.js | index.html | package-lock.json

02-express-tutorial > final > JS 07-params-query.js > app.get('/api/products/:productID') callback

```
1   const express = require('express')
2   const app = express()
3   const { products } = require('./data')
4
5   app.get('/', (req, res) => {
6     res.send('<h1> Home Page</h1><a href="/api/products">products</a>')
7   })
8   app.get('/api/products', (req, res) => {
9     const newProducts = products.map((product) => {
10      const { id, name, image } = product
11      return { id, name, image }
12    })
13
14    res.json(newProducts)
15  })
16  app.get('/api/products/:productID', (req, res) => {
17    // console.log(req)
18    // console.log(req.params)
19    const { productID } = req.params
20
21    const singleProduct = products.find(
22      (product) => product.id === Number(productID)
23    )
24    if (!singleProduct) {
25      return res.status(404).send('Product Does Not Exist')
26    }
27
28    return res.json(singleProduct)
29  })
30
31  app.get('/api/products/:productID/reviews/:reviewID', (req, res) => {
32    console.log(req.params)
33    res.send('hello world')
34  })
```

route parameter

multiple params

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL — node - final

```
    at Function.executeUserEntryPoint [as runMain] (node:internal/modules/run_main:81:12) {
  code: 'MODULE_NOT_FOUND',
  requireStack: [
    '/Users/harshali15/Desktop/node-express-course/02-express-tutorial/final/06-basic-json.js'
  ]
}
harshali15@Harshalis-Air final % node 06-basic-json.js
Server is listening on port 5000....
^C
harshali15@Harshalis-Air final % node 07-params-query.js
Server is listening on port 5000....
```

Ln 26, Col 4 Spaces: 2 UTF-8 LF {} Jav

MacBook Air

JS 05-all-static.js JS 06-basic-json.js JS 07-params-query.js × JS data.js <> index.html {} package-lock.json

02-express-tutorial > final > JS 07-params-query.js > app.get('/api/products/:productID') callback

```
30
31  app.get('/api/products/:productID/reviews/:reviewID', (req, res) => {
32    console.log(req.params)
33    res.send('hello world')
34  })
35
36  app.get('/api/v1/query', (req, res) => {
37    // console.log(req.query)
38    const { search, limit } = req.query
39    let sortedProducts = [...products]
40
41    if (search) {
42      sortedProducts = sortedProducts.filter((product) => {
43        return product.name.startsWith(search)
44      })
45    }
46    if (limit) {
47      sortedProducts = sortedProducts.slice(0, Number(limit))
48    }
49    if (sortedProducts.length < 1) {
50      // res.status(200).send('no products matched your search');
51      return res.status(200).json({ sucess: true, data: [] })
52    }
53    res.status(200).json(sortedProducts)
54  })
55
56  app.listen(5000, () => {
57    console.log('Server is listening on port 5000....')
58  })
59
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL node - final

```
    at Function.executeUserEntryPoint [as runMain] (node:internal/modules/run_main:81:12) {
  code: 'MODULE_NOT_FOUND',
  requireStack: [
    '/Users/harshali15/Desktop/node-express-course/02-express-tutorial/final/06-basic-json.js'
  ]
}
harshali15@Harshalis-Air final % node 06-basic-json.js
Server is listening on port 5000....
^C
harshali15@Harshalis-Air final % node 07-params-query.js
Server is listening on port 5000....
```

Ln 26, Col 4 Spaces: 2 UTF-8 LF {} JavaScr

MacBook Air

Middleware — functions that execute during request to the server
Each middleware fn has access to request & response.

req ⇒ middleware ⇒ res.

```
const app = express()
const logger = (req, res, next) => {
  const method = req.method
  const url = req.url
  const time = new Date().getFullYear()
  console.log(url, method, time)
  next()
}
```

args are passed by express automatically.

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

Better way

put logger in a separate file logger.js.

```
const logger = require('./logger')
app.use(logger)
```
— log for each url

OR

```
app.use('/api', logger)
```
→ log for any url starting with /api.

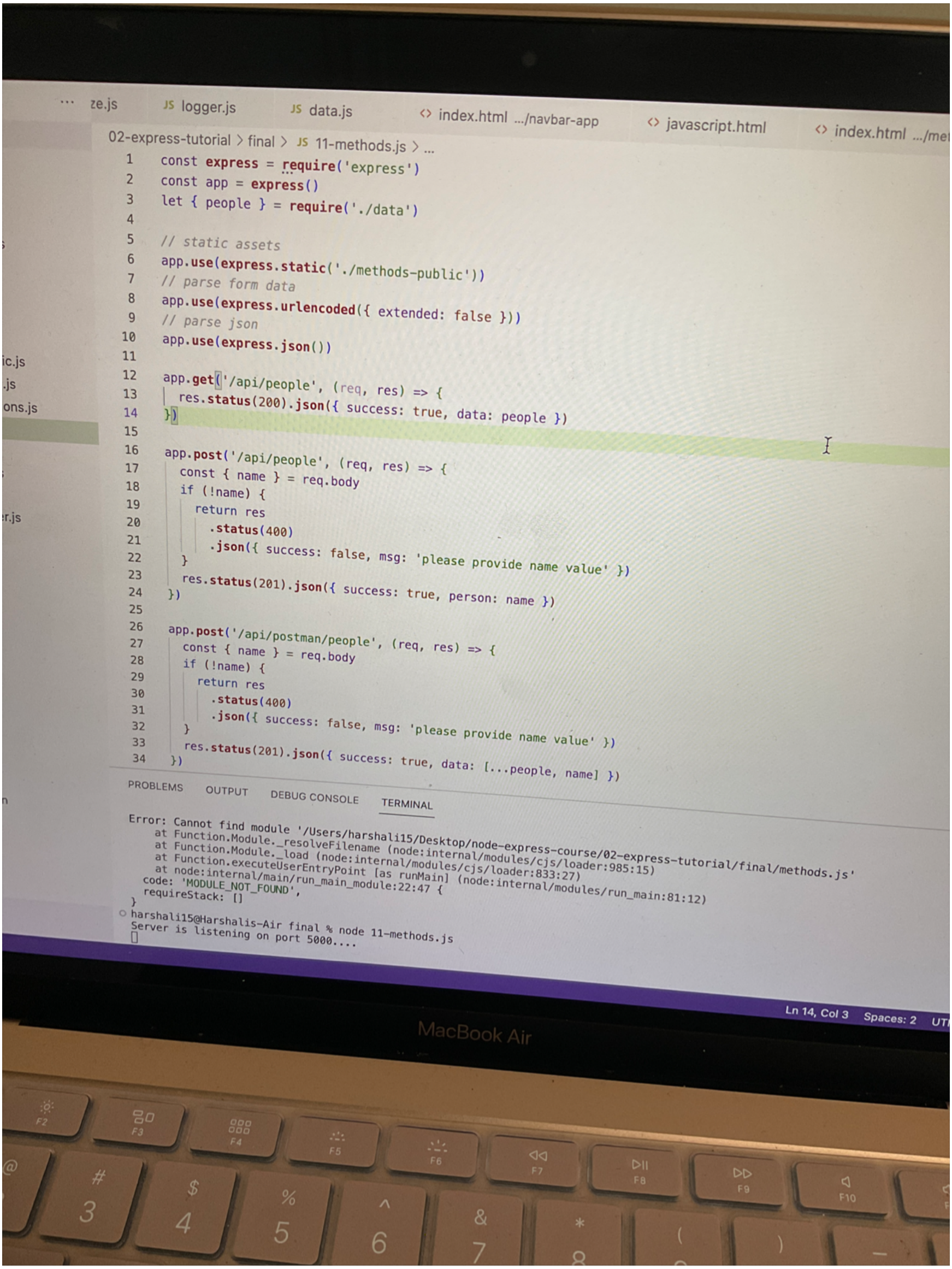
OR

for multiple fns

```
app.use([logger, authorize])
```
executed in order.

Logger-
1) your own
2) express (static)
3) 3rd party (morgan('tiny'))

Other http methods


ze.js
logger.js
data.js
index.html .../navbar-app
javascript.html
index.html .../me
02-express-tutorial > final > 11-methods.js > ...
const express = require('express')
const app = express()
let { people } = require('./data')
// static assets
app.use(express.static('./methods-public'))
// parse form data
app.use(express.urlencoded({ extended: false }))
// parse json
app.use(express.json())
app.get('/api/people', (req, res) => {
res.status(200).json({ success: true, data: people })
})
app.post('/api/people', (req, res) => {
const { name } = req.body
if (!name) {
return res
.status(400)
.json({ success: false, msg: 'please provide name value' })
}
res.status(201).json({ success: true, person: name })
})
app.post('/api/postman/people', (req, res) => {
const { name } = req.body
if (!name) {
return res
.status(400)
.json({ success: false, msg: 'please provide name value' })
}
res.status(201).json({ success: true, data: [...people, name] })
})
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
Error: Cannot find module '/Users/harshali15/Desktop/node-express-course/02-express-tutorial/final/methods.js'
at Function.Module._resolveFilename (node:internal/modules/cjs/loader:985:15)
at Function.Module._load (node:internal/modules/cjs/loader:833:27)
at node:internal/main/run_main_module:22:47 {
code: 'MODULE_NOT_FOUND',
requireStack: []
}
harshali15@Harshalis-Air final % node 11-methods.js
Server is listening on port 5000....
Ln 14, Col 3
Spaces: 2
MacBook Air

ze.js | logger.js | data.js | index.html .../navbar-app | javascript.html | index.html .../methods-pub

02-express-tutorial > final > 11-methods.js > ...

```
36  app.post('/login', (req, res) => {
37    const { name } = req.body
38    if (name) {
39      return res.status(200).send(`Welcome ${name}`)
40    }
41
42    res.status(401).send('Please Provide Credentials')
43  })
44
45  app.put('/api/people/:id', (req, res) => {
46    const { id } = req.params
47    const { name } = req.body
48
49    const person = people.find((person) => person.id === Number(id))
50
51    if (!person) {
52      return res
53        .status(404)
54        .json({ success: false, msg: `no person with id ${id}` })
55    }
56    const newPeople = people.map((person) => {
57      if (person.id === Number(id)) {
58        person.name = name
59      }
60      return person
61    })
62    res.status(200).json({ success: true, data: newPeople })
63  })
64
65  app.delete('/api/people/:id', (req, res) => {
66    const person = people.find((person) => person.id === Number(req.params.id))
67    if (!person) {
68      return res
69        .status(404)
```

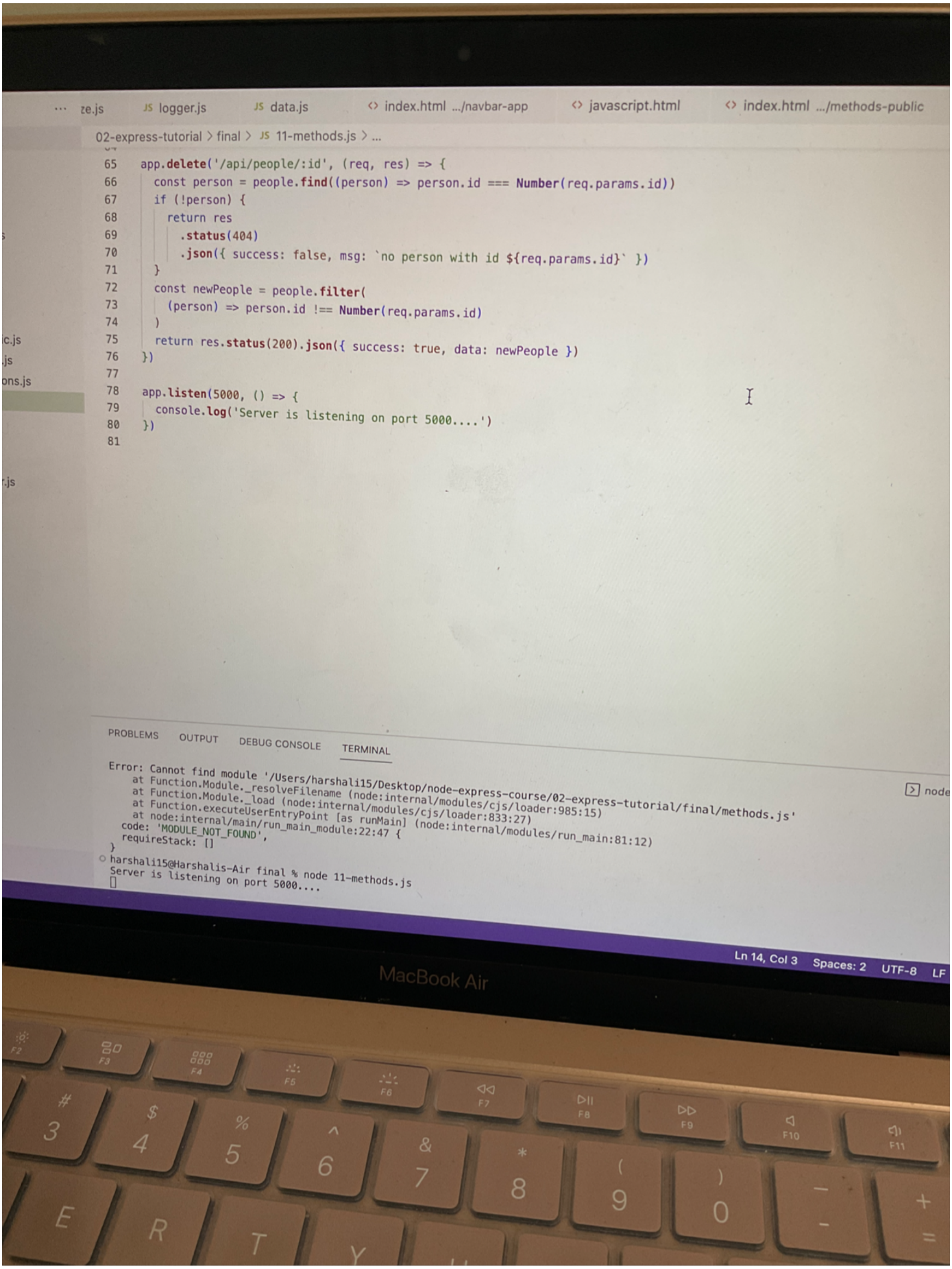
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
Error: Cannot find module '/Users/harshali15/Desktop/node-express-course/02-express-tutorial/final/methods.js'
    at Function.Module._resolveFilename (node:internal/modules/cjs/loader:985:15)
    at Function.Module._load (node:internal/modules/cjs/loader:833:27)
    at Function.executeUserEntryPoint [as runMain] (node:internal/modules/run_main:81:12)
    at node:internal/main/run_main_module:22:47 {
  code: 'MODULE_NOT_FOUND',
  requireStack: []
}
harshali15@Harshalis-Air final % node 11-methods.js
Server is listening on port 5000....
```

Ln 14, Col 3 Spaces: 2 UT

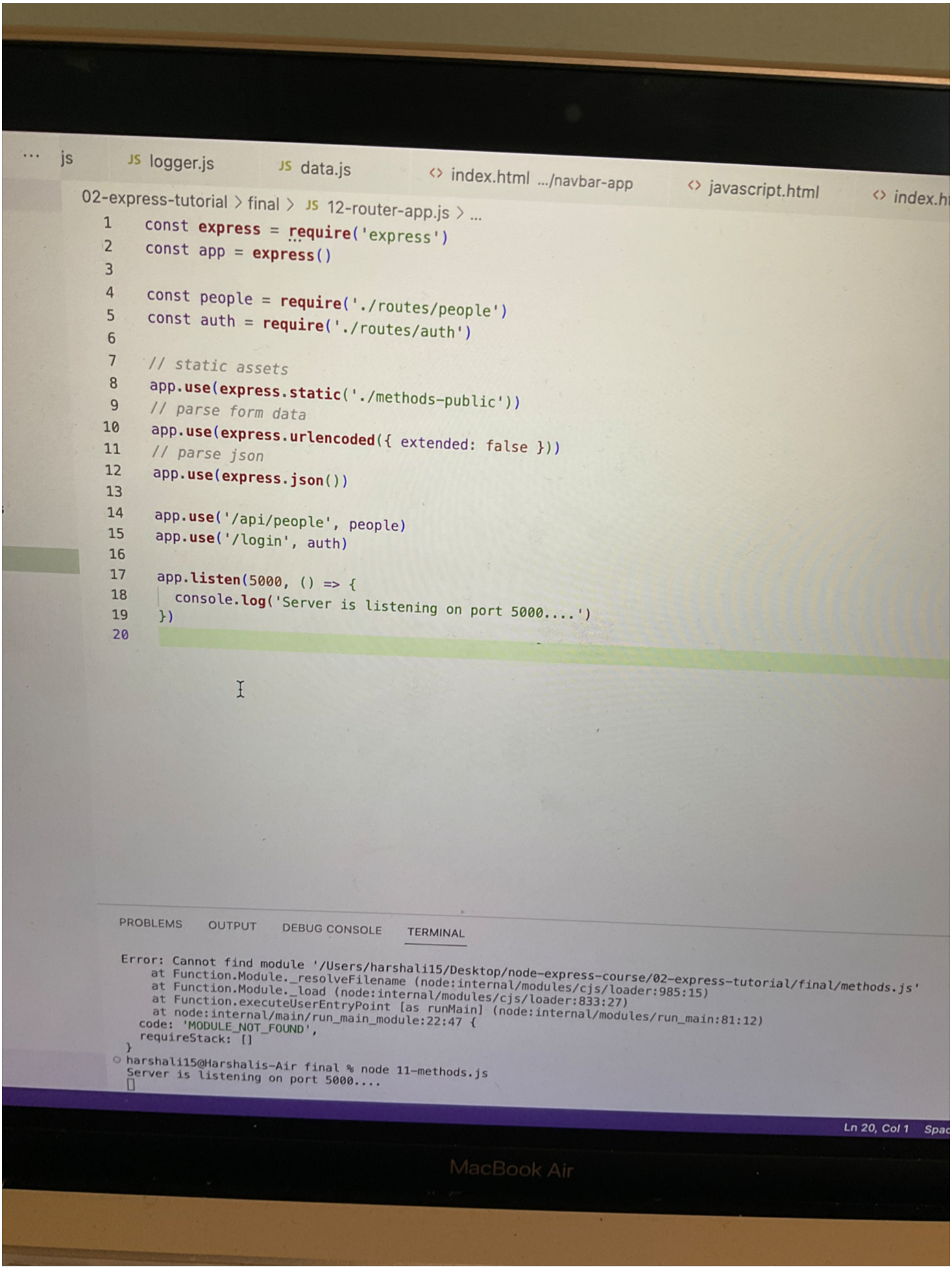
MacBook Air

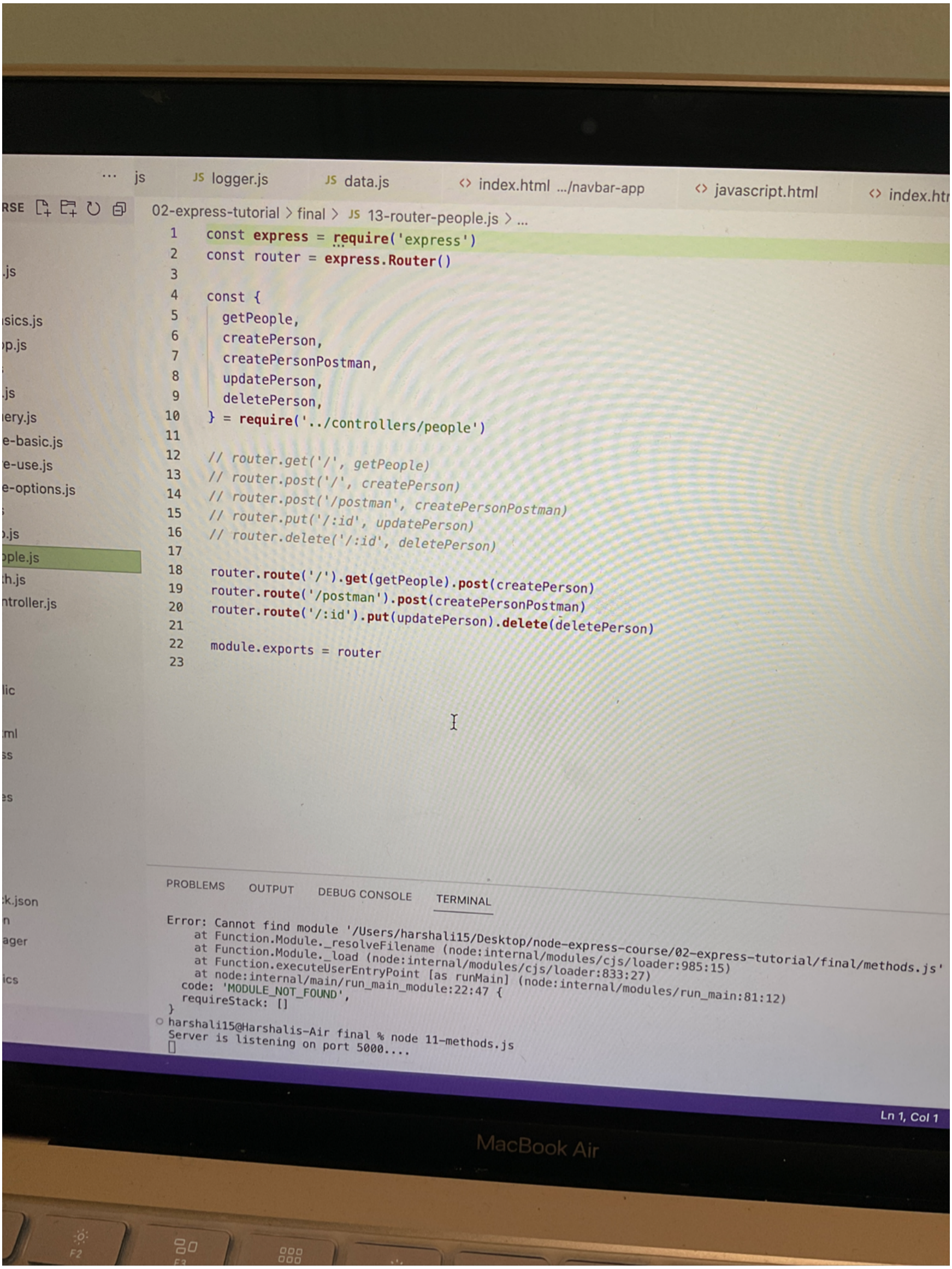
F3 F4 F5 F6 F7 F8 F9 F10

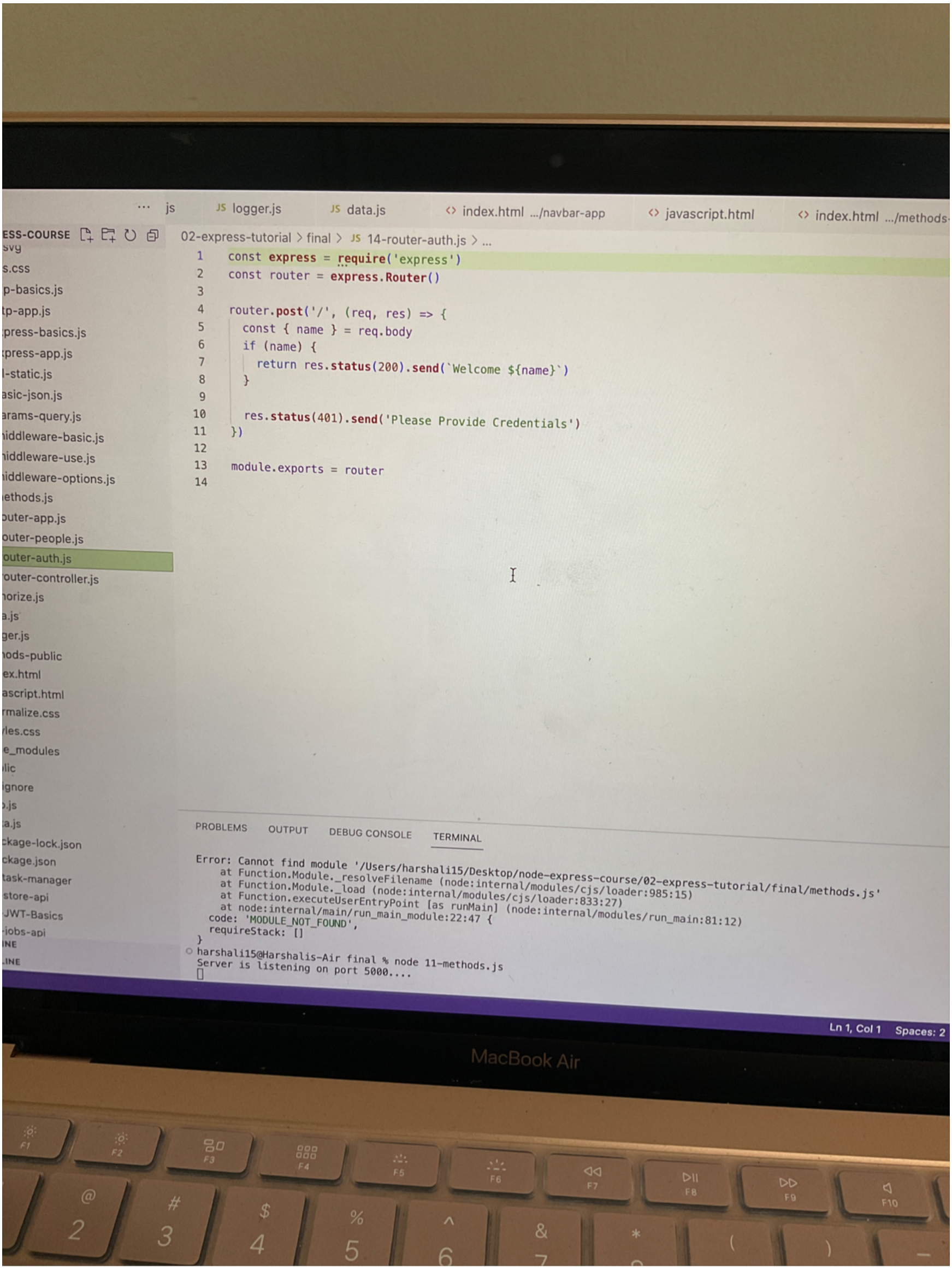


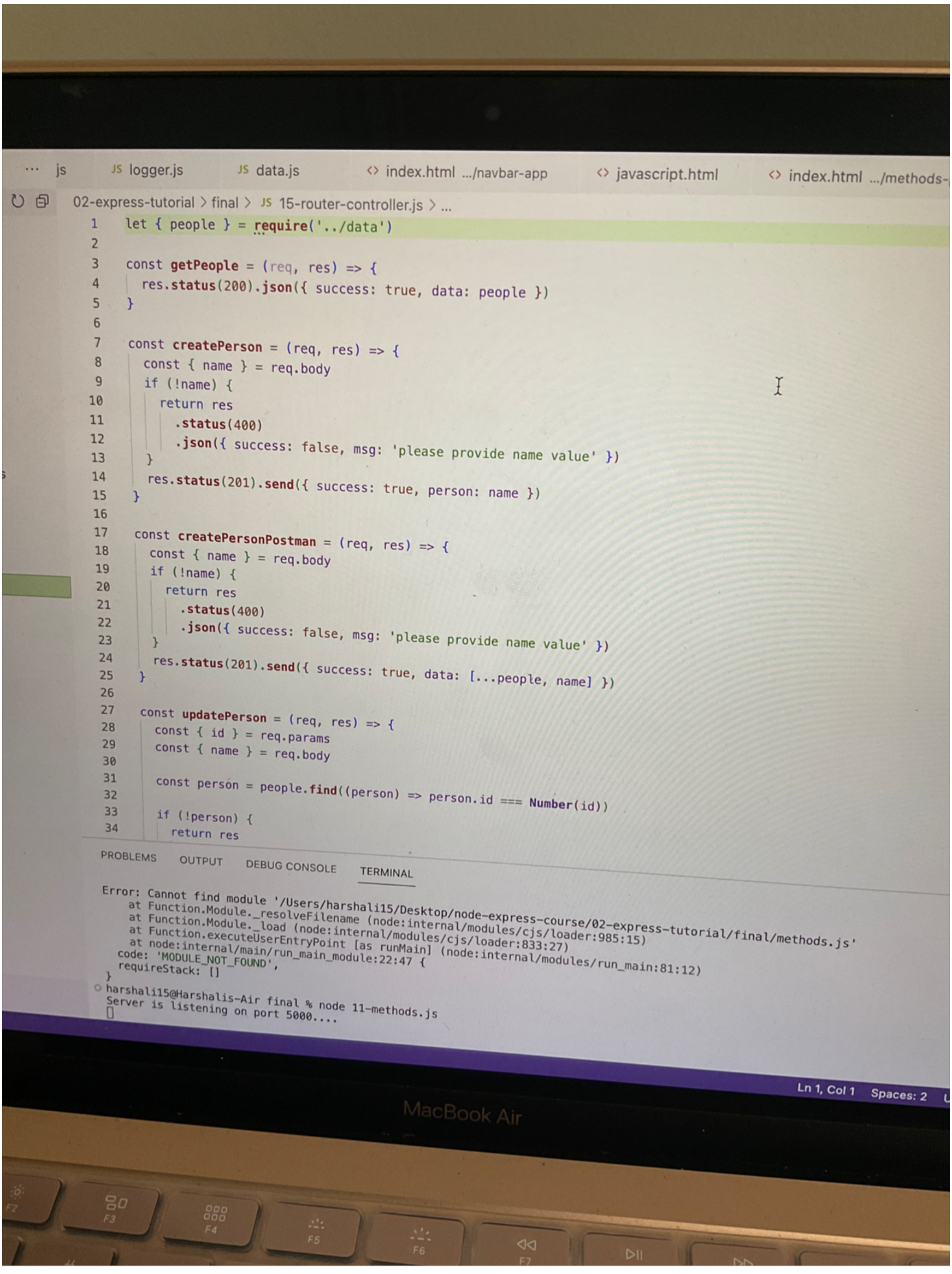
There is certain business in the code
so we use router

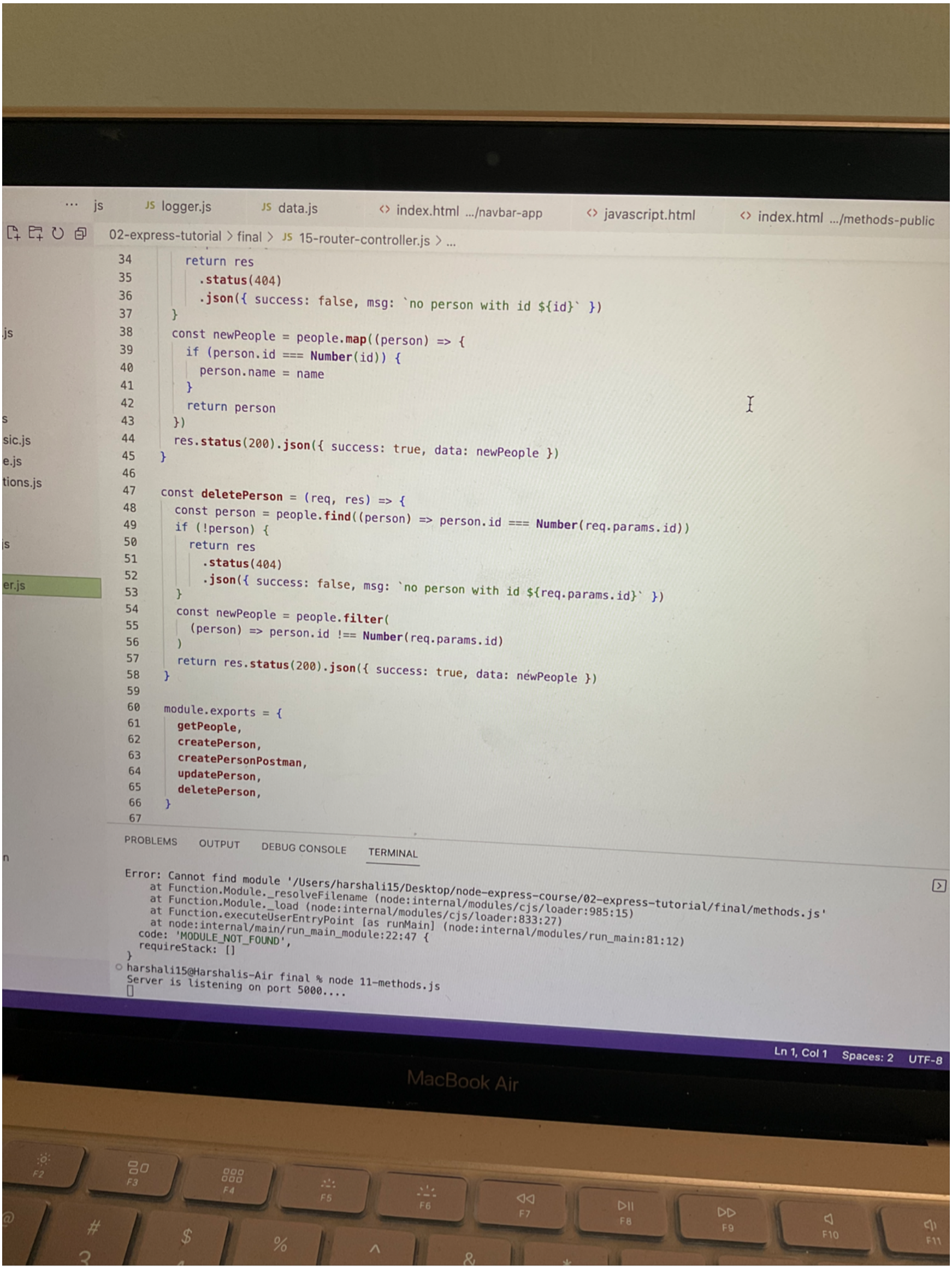
```
const router = express.Router()
```


js
logger.js
data.js
index.html .../navbar-app
javascript.html
index.h
02-express-tutorial > final > 12-router-app.js > ...
1 const express = require('express')
2 const app = express()
3
4 const people = require('./routes/people')
5 const auth = require('./routes/auth')
6
7 // static assets
8 app.use(express.static('./methods-public'))
9 // parse form data
10 app.use(express.urlencoded({ extended: false }))
11 // parse json
12 app.use(express.json())
13
14 app.use('/api/people', people)
15 app.use('/login', auth)
16
17 app.listen(5000, () => {
18 console.log('Server is listening on port 5000....')
19 })
20
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
Error: Cannot find module '/Users/harshali15/Desktop/node-express-course/02-express-tutorial/final/methods.js'
at Function.Module._resolveFilename (node:internal/modules/cjs/loader:985:15)
at Function.Module._load (node:internal/modules/cjs/loader:833:27)
at Function.executeUserEntryPoint [as runMain] (node:internal/modules/run_main:81:12)
at node:internal/main/run_main_module:22:47 {
code: 'MODULE_NOT_FOUND',
requireStack: []
}
harshali15@Harshalis-Air final % node 11-methods.js
Server is listening on port 5000....
Ln 20, Col 1
MacBook Air


js
logger.js
data.js
index.html .../navbar-app
javascript.html
index.htr
02-express-tutorial > final > 13-router-people.js > ...
1 const express = require('express')
2 const router = express.Router()
3
4 const {
5 getPeople,
6 createPerson,
7 createPersonPostman,
8 updatePerson,
9 deletePerson,
10 } = require('../controllers/people')
11
12 // router.get('/', getPeople)
13 // router.post('/', createPerson)
14 // router.post('/postman', createPersonPostman)
15 // router.put('/:id', updatePerson)
16 // router.delete('/:id', deletePerson)
17
18 router.route('/').get(getPeople).post(createPerson)
19 router.route('/postman').post(createPersonPostman)
20 router.route('/:id').put(updatePerson).delete(deletePerson)
21
22 module.exports = router
23
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
Error: Cannot find module '/Users/harshali15/Desktop/node-express-course/02-express-tutorial/final/methods.js'
at Function.Module._resolveFilename (node:internal/modules/cjs/loader:985:15)
at Function.Module._load (node:internal/modules/cjs/loader:833:27)
at node:internal/main/run_main_module:22:47 {
code: 'MODULE_NOT_FOUND',
requireStack: []
}
harshali15@Harshalis-Air final % node 11-methods.js
Server is listening on port 5000....
Ln 1, Col 1
MacBook Air


js
logger.js
data.js
index.html .../navbar-app
javascript.html
index.html .../methods-
ESS-COURSE
02-express-tutorial > final > 14-router-auth.js > ...
s.css
p-basics.js
tp-app.js
press-basics.js
press-app.js
l-static.js
asic-json.js
arams-query.js
iddleware-basic.js
iddleware-use.js
iddleware-options.js
ethods.js
outer-app.js
outer-people.js
outer-auth.js
outer-controller.js
norize.js
a.js
ger.js
nods-public
ex.html
ascript.html
rmalize.css
yles.css
e_modules
lic
gnore
.js
a.js
ckage-lock.json
ckage.json
task-manager
store-api
JWT-Basics
jobs-api
INE
LINE
1 const express = require('express')
2 const router = express.Router()
3
4 router.post('/', (req, res) => {
5 const { name } = req.body
6 if (name) {
7 return res.status(200).send(`Welcome ${name}`)
8 }
9
10 res.status(401).send('Please Provide Credentials')
11 })
12
13 module.exports = router
14
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
Error: Cannot find module '/Users/harshali15/Desktop/node-express-course/02-express-tutorial/final/methods.js'
at Function.Module._resolveFilename (node:internal/modules/cjs/loader:985:15)
at Function.Module._load (node:internal/modules/cjs/loader:833:27)
at Function.executeUserEntryPoint [as runMain] (node:internal/modules/run_main:81:12)
at node:internal/main/run_main_module:22:47 {
code: 'MODULE_NOT_FOUND',
requireStack: []
}
harshali15@Harshalis-Air final % node 11-methods.js
Server is listening on port 5000....
Ln 1, Col 1
Spaces: 2
MacBook Air


js
logger.js
data.js
index.html .../navbar-app
javascript.html
index.html .../methods-
02-express-tutorial > final > 15-router-controller.js > ...
let { people } = require('../data')
const getPeople = (req, res) => {
res.status(200).json({ success: true, data: people })
}
const createPerson = (req, res) => {
const { name } = req.body
if (!name) {
return res
.status(400)
.json({ success: false, msg: 'please provide name value' })
}
res.status(201).send({ success: true, person: name })
}
const createPersonPostman = (req, res) => {
const { name } = req.body
if (!name) {
return res
.status(400)
.json({ success: false, msg: 'please provide name value' })
}
res.status(201).send({ success: true, data: [...people, name] })
}
const updatePerson = (req, res) => {
const { id } = req.params
const { name } = req.body
const person = people.find((person) => person.id === Number(id))
if (!person) {
return res
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
Error: Cannot find module '/Users/harshali15/Desktop/node-express-course/02-express-tutorial/final/methods.js'
at Function.Module._resolveFilename (node:internal/modules/cjs/loader:985:15)
at Function.Module._load (node:internal/modules/cjs/loader:833:27)
at node:internal/main/run_main_module:22:47 {
code: 'MODULE_NOT_FOUND',
requireStack: []
}
harshali15@Harshalis-Air final % node 11-methods.js
Server is listening on port 5000....
Ln 1, Col 1
Spaces: 2
MacBook Air
F2
F3
F4
F5
F6
F7


js
logger.js
data.js
index.html .../navbar-app
javascript.html
index.html .../methods-public
02-express-tutorial > final > 15-router-controller.js > ...
return res
.status(404)
.json({ success: false, msg: `no person with id ${id}` })
}
const newPeople = people.map((person) => {
if (person.id === Number(id)) {
person.name = name
}
return person
})
res.status(200).json({ success: true, data: newPeople })
}
const deletePerson = (req, res) => {
const person = people.find((person) => person.id === Number(req.params.id))
if (!person) {
return res
.status(404)
.json({ success: false, msg: `no person with id ${req.params.id}` })
}
const newPeople = people.filter(
(person) => person.id !== Number(req.params.id)
)
return res.status(200).json({ success: true, data: newPeople })
}
module.exports = {
getPeople,
createPerson,
createPersonPostman,
updatePerson,
deletePerson,
}
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
Error: Cannot find module '/Users/harshali15/Desktop/node-express-course/02-express-tutorial/final/methods.js'
at Function.Module._resolveFilename (node:internal/modules/cjs/loader:985:15)
at Function.Module._load (node:internal/modules/cjs/loader:833:27)
at Function.executeUserEntryPoint [as runMain] (node:internal/modules/run_main:81:12)
code: 'MODULE_NOT_FOUND',
requireStack: []
}
harshali15@Harshalis-Air final % node 11-methods.js
Server is listening on port 5000....
Ln 1, Col 1
Spaces: 2
UTF-8
MacBook Air