

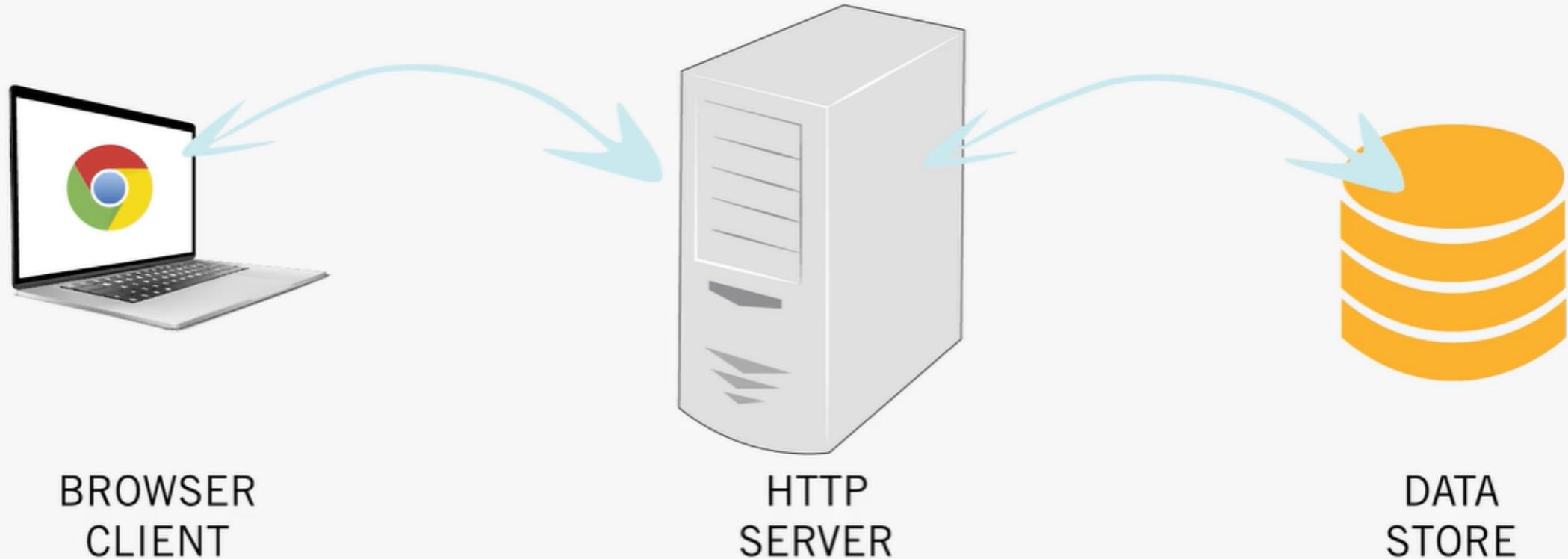


Professional Certificate in Coding: Full Stack Development with MERN: Week 21

---

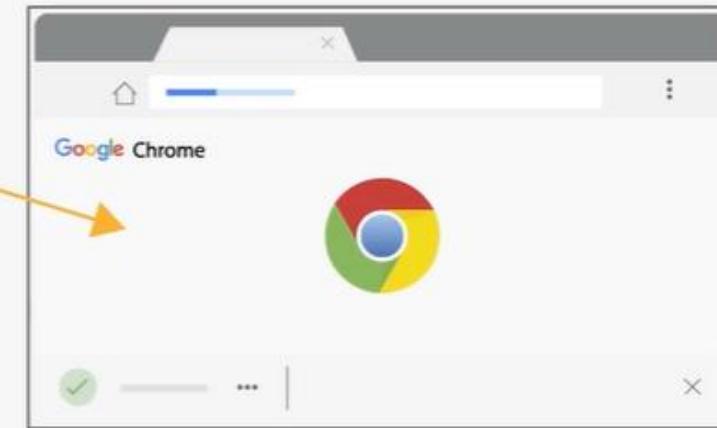
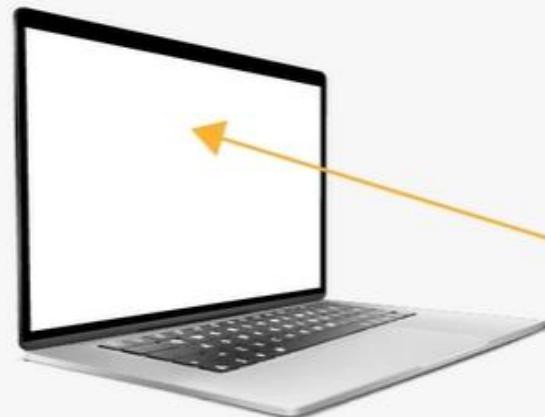
## Introduction To Tiered Applications

# Three Tiers: Client, Server, & Storage



# Little Web Server

Express  
  

# Creating Little Server (1/20)



```
✓ Desktop % mkdir littleserver
✓ Desktop % cd littleserver
✓ littleserver % npm init
```

A screenshot of a macOS terminal window. The window title is "littleserver". The terminal prompt is "littleserver %". The user has run three commands: "mkdir littleserver", "cd littleserver", and "npm init". The "npm init" command is still being typed, with the cursor visible at the end of the line.

## Creating Little Server (2/20)

```
e defaults.

See `npm help init` for definitive documentation on these fields
and exactly what they do.

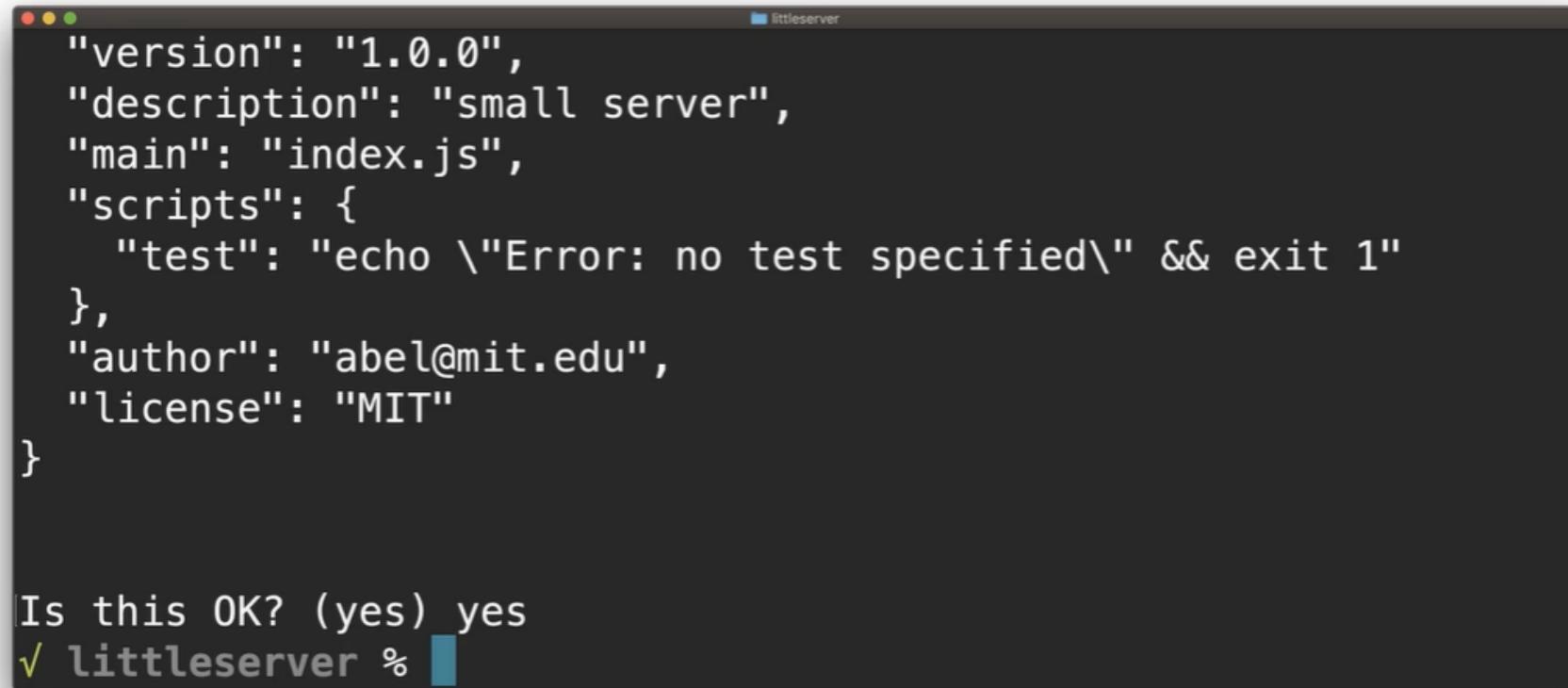
Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (littleserver)
version: (1.0.0)
description: small server
entry point: (index.js) █
```

## Creating Little Server (3/20)

```
Use `npm install <pkg>` afterwards to install a package and  
save it as a dependency in the package.json file.  
  
Press ^C at any time to quit.  
package name: (littleserver)  
version: (1.0.0)  
description: small server  
entry point: (index.js)  
test command:  
git repository:  
keywords:  
author: abel@mit.edu  
license: (ISC) MIT
```

## Creating Little Server (4/20)



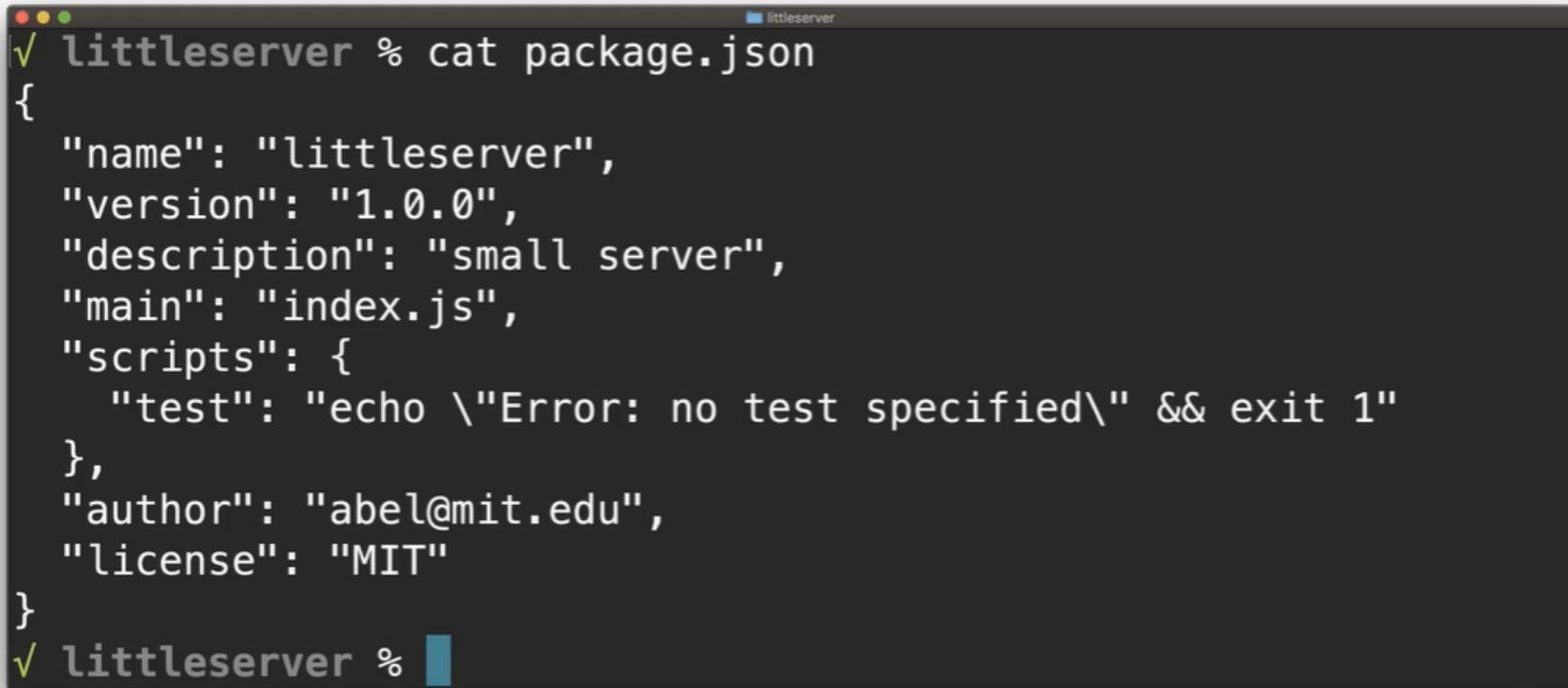
A screenshot of a terminal window titled "littleserver". The window contains the following JSON code:

```
"version": "1.0.0",
"description": "small server",
"main": "index.js",
"scripts": {
  "test": "echo \\\"Error: no test specified\\\" && exit 1"
},
"author": "abel@mit.edu",
"license": "MIT"
}
```

Below the code, the terminal prompt shows:

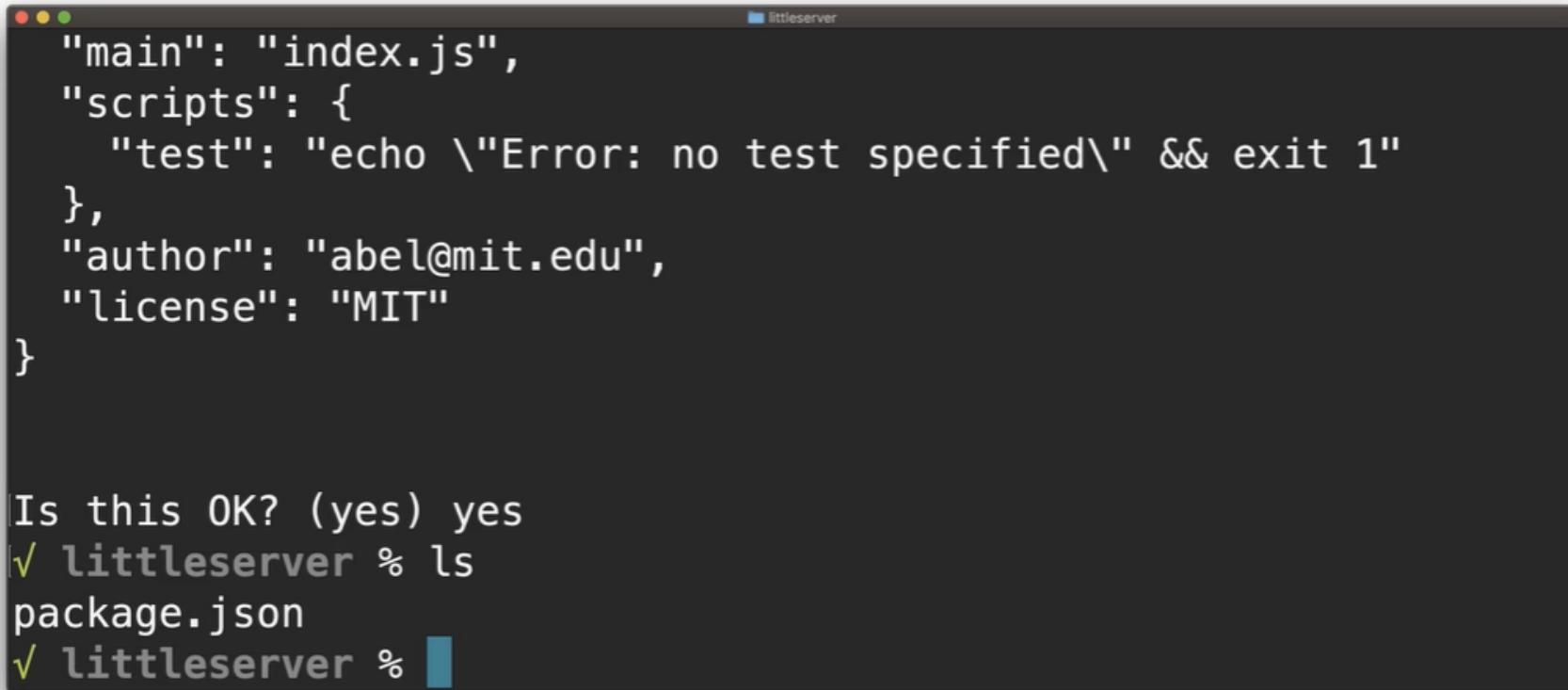
```
Is this OK? (yes) yes
✓ littleserver %
```

## Creating Little Server (5/20)



```
littleserver % cat package.json
{
  "name": "littleserver",
  "version": "1.0.0",
  "description": "small server",
  "main": "index.js",
  "scripts": {
    "test": "echo \\\"Error: no test specified\\\" && exit 1"
  },
  "author": "abel@mit.edu",
  "license": "MIT"
}
littleserver %
```

## Creating Little Server (6/20)



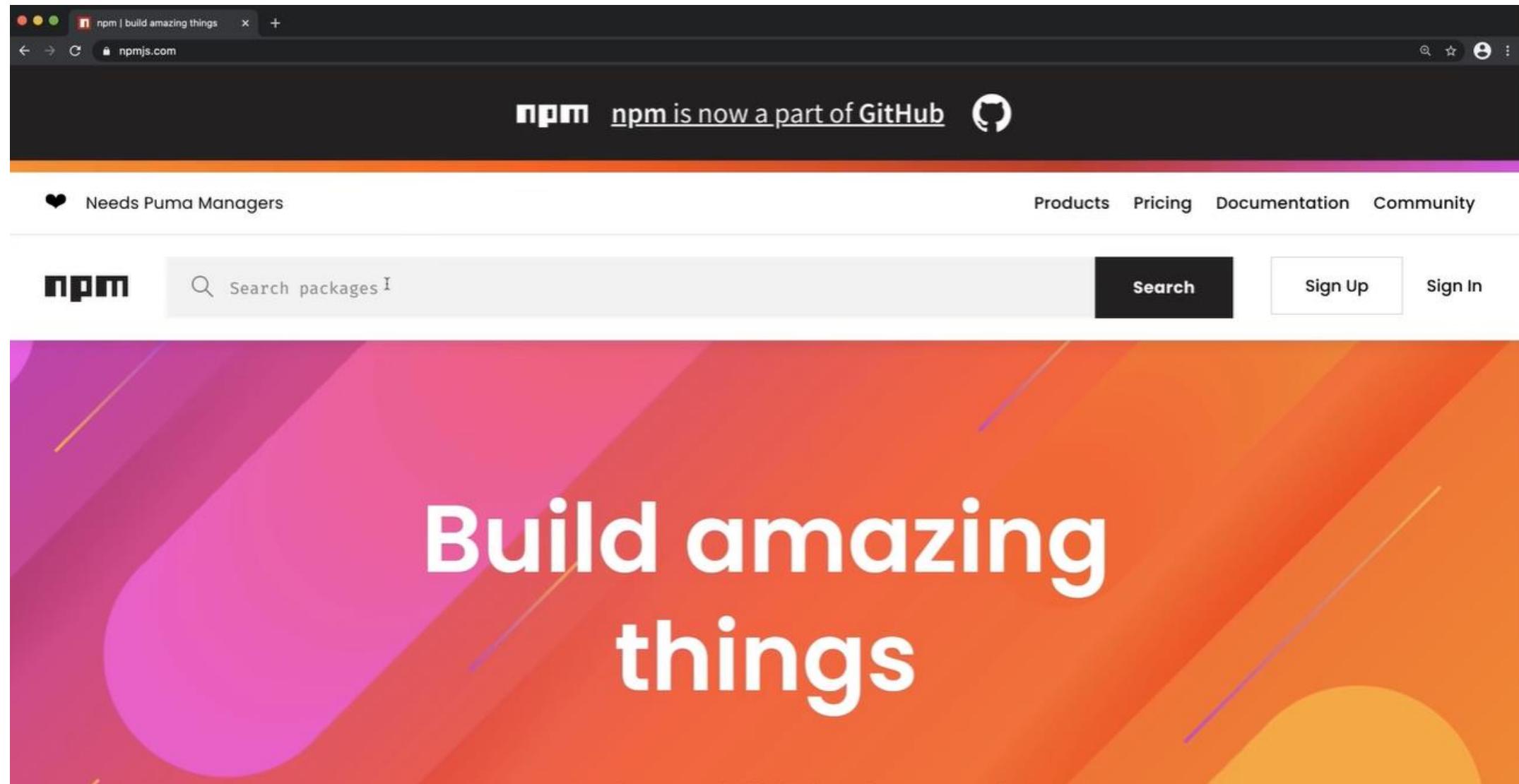
```
"main": "index.js",
"scripts": {
  "test": "echo \\\"Error: no test specified\\\" && exit 1"
},
"author": "abel@mit.edu",
"license": "MIT"
}

Is this OK? (yes) yes
└─ littleserver % ls
  package.json
└─ littleserver %
```

## Creating Little Server (7/20)

```
✓ littleserver % cat package.json
{
  "name": "littleserver",
  "version": "1.0.0",
  "description": "small server",
  "main": "index.js",
  "scripts": {
    "test": "echo \\\"Error: no test specified\\\" && exit 1"
  },
  "author": "abel@mit.edu",
  "license": "MIT"
}
✓ littleserver %
```

# Creating Little Server (7/20)



# Creating Little Server (8/20)

express - npm search

npm is now a part of GitHub

Nicely Packaged Make

Products Pricing Documentation Community

Sign Up Sign In

npm express

Search

21589 packages found

Sort Packages

Optimal

Popularity

Quality

Maintenance

express exact match

Fast, unopinionated, minimalist web framework

p  
q  
m

express framework sinatra web rest restful router app api

dougwilson published 4.17.1 • a year ago

path-to-regexp

Express style path to RegExp utility

express regexp route routing

p  
q  
m

blakeembrey published 6.2.0 • 14 days ago

The screenshot shows a search results page for the npm package 'express'. At the top, there's a banner stating 'npm is now a part of GitHub'. Below the banner, there are navigation links for 'Products', 'Pricing', 'Documentation', and 'Community'. On the left, there's a sidebar with 'Sort Packages' options: 'Optimal', 'Popularity', 'Quality', and 'Maintenance'. The main search results area shows the first result for 'express', which is described as a 'Fast, unopinionated, minimalist web framework'. It includes a link to the package page, the author's profile (dougwilson), the publication date (4.17.1 published a year ago), and a list of tags: express, framework, sinatra, web, rest, restful, router, app, api. Below this, another package, 'path-to-regexp', is listed with its description ('Express style path to RegExp utility'), author (blakeembrey), publication date (6.2.0 published 14 days ago), and tags: express, regexp, route, routing.

# Creating Little Server (9/20)

express - npm

npmjs.com/package/express

express

4.17.1 • Public • Published a year ago

Readme Explore BETA 30 Dependencies 46,655 Dependents 264 Versions

# express

Fast, unopinionated, minimalist web framework for [node](#).

npm v4.17.1 downloads 59M/month linux passing windows passing coverage 100%

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

app.get('/', function (req, res) {
  res.send('Hello World')
})

app.listen(3000)
```

Install

> npm i express

Copy Command to Clipboard

Weekly Downloads

14,057,862

Version 4.17.1 License MIT

Unpacked Size 208 kB Total Files 16

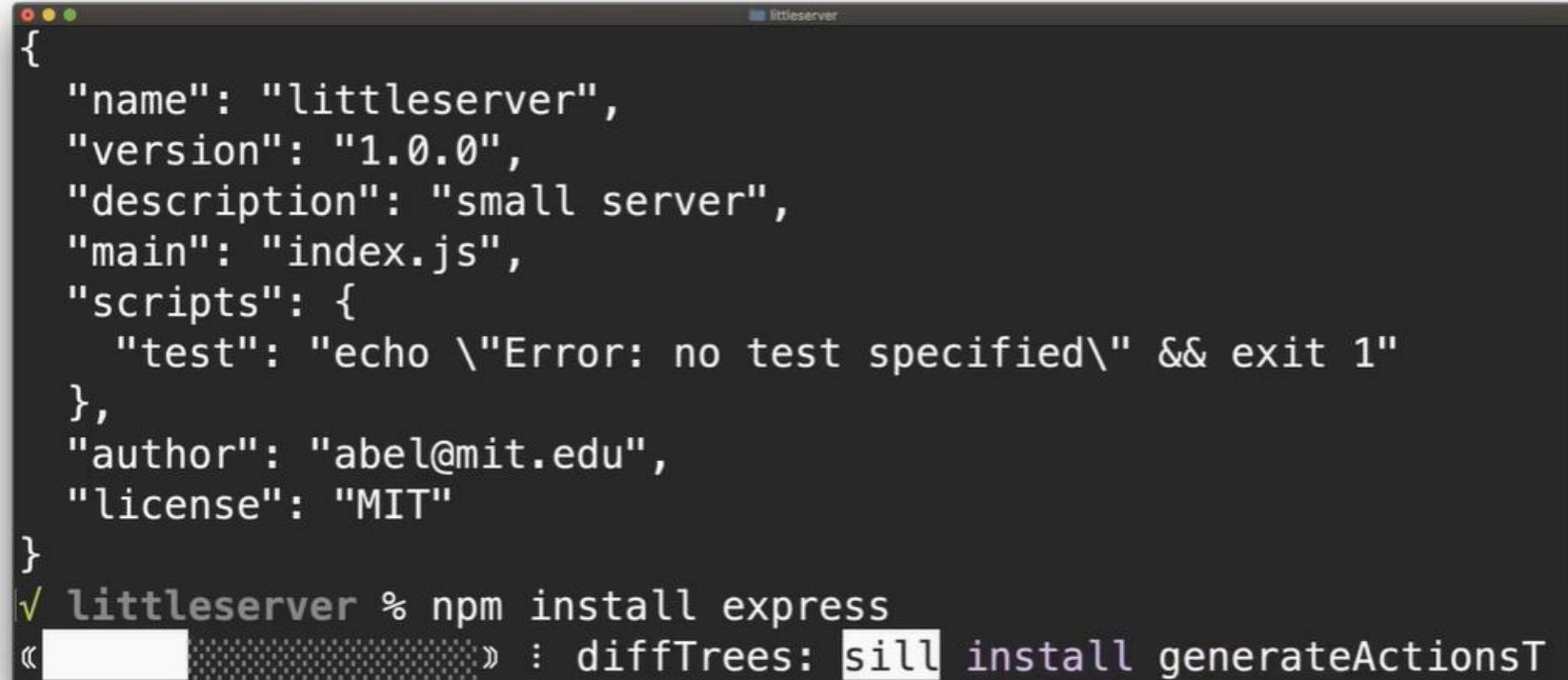
Issues 97 Pull Requests 52

Homepage [expressjs.com/](http://expressjs.com/)

Repository

## Installation

# Creating Little Server (10/20)



```
{  
  "name": "littleserver",  
  "version": "1.0.0",  
  "description": "small server",  
  "main": "index.js",  
  "scripts": {  
    "test": "echo \\\"Error: no test specified\\\" && exit 1"  
  },  
  "author": "abel@mit.edu",  
  "license": "MIT"  
}  
littleserver % npm install express  
» : diffTrees: sill install generateActionsT
```

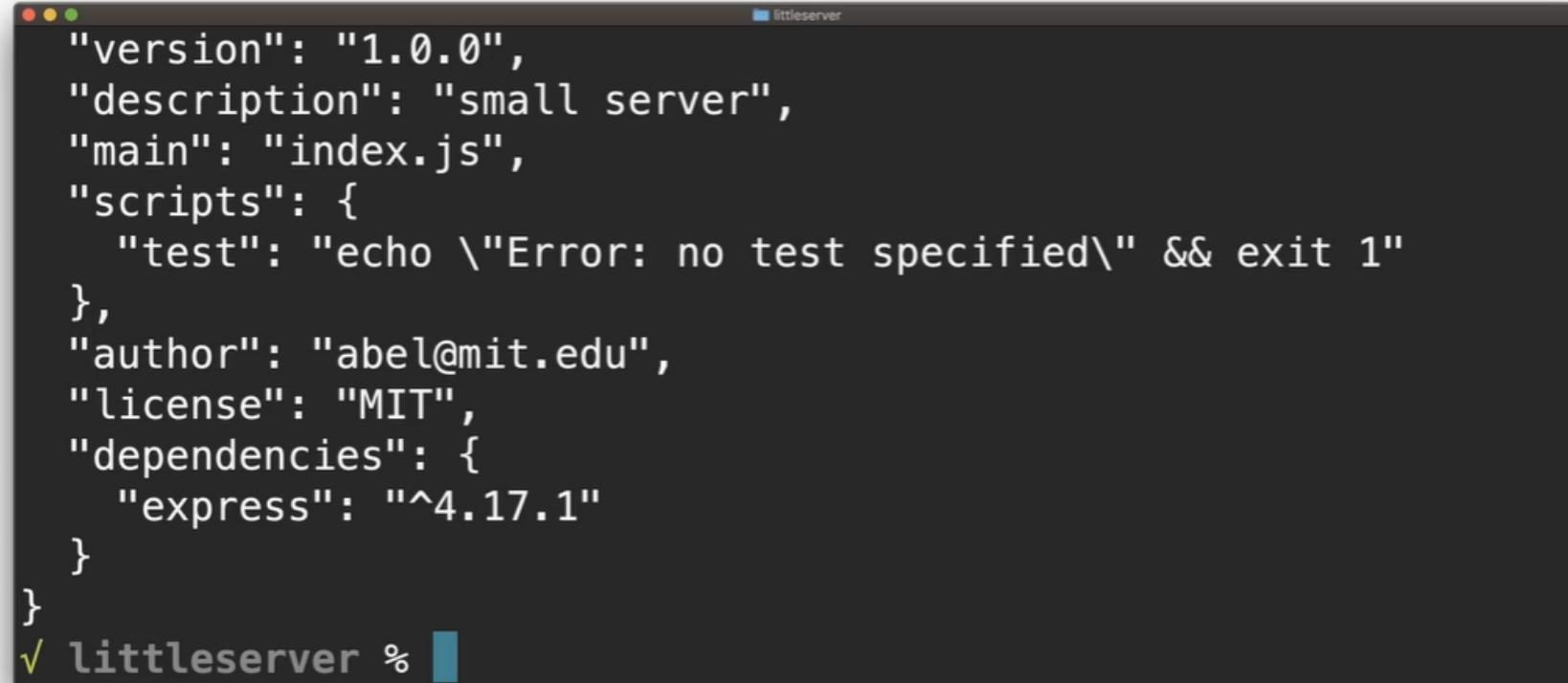
# Creating Little Server (11/20)

```
  "license": "MIT"
}
✓ littleserver % npm install express
npm notice created a lockfile as package-lock.json. You should c
ommit this file.
npm WARN littleserver@1.0.0 No repository field.

+ express@4.17.1
added 50 packages from 37 contributors and audited 50 packages i
n 1.407s
found 0 vulnerabilities

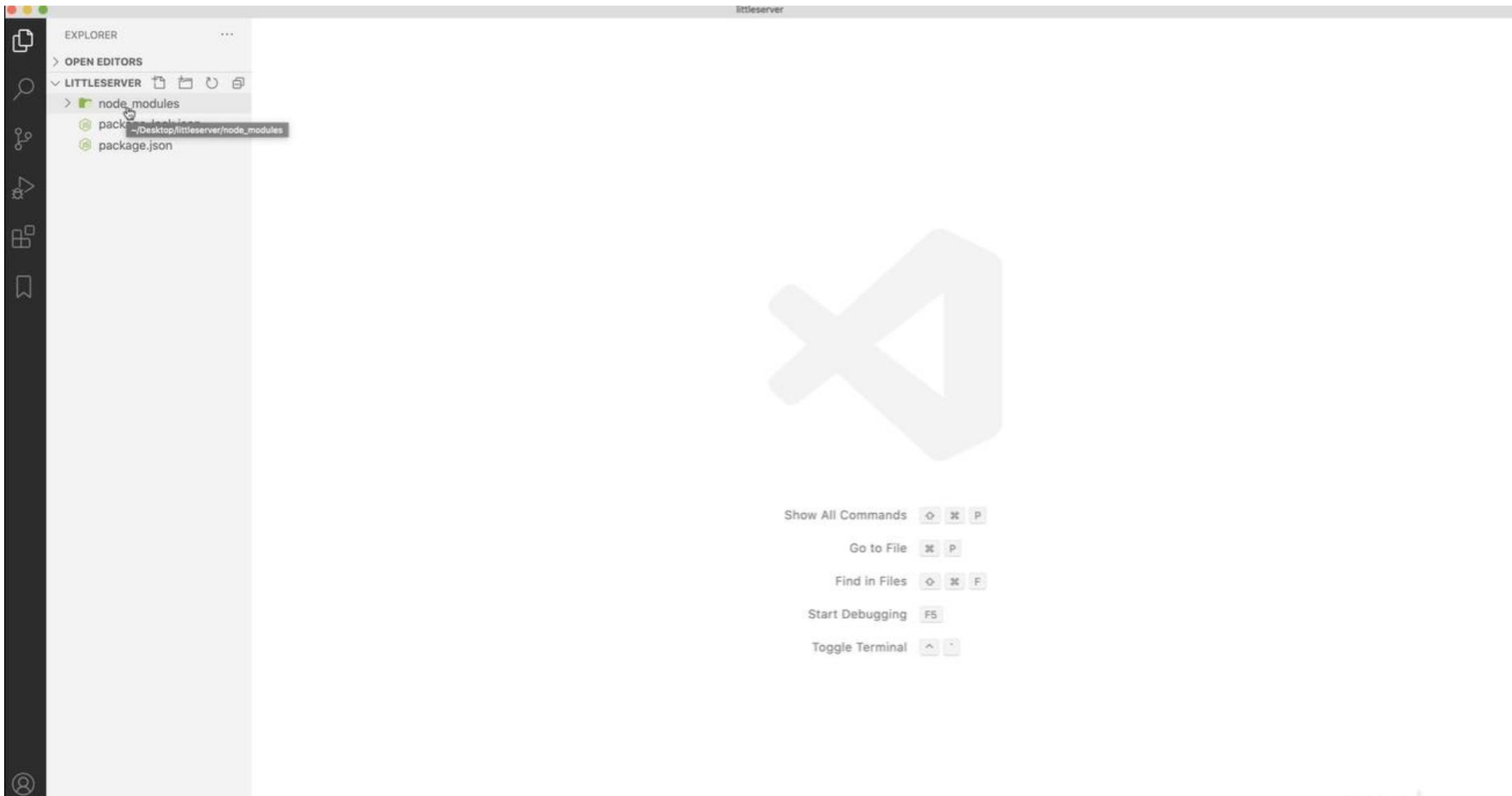
✓ littleserver % cat package.json
```

# Creating Little Server (12/20)

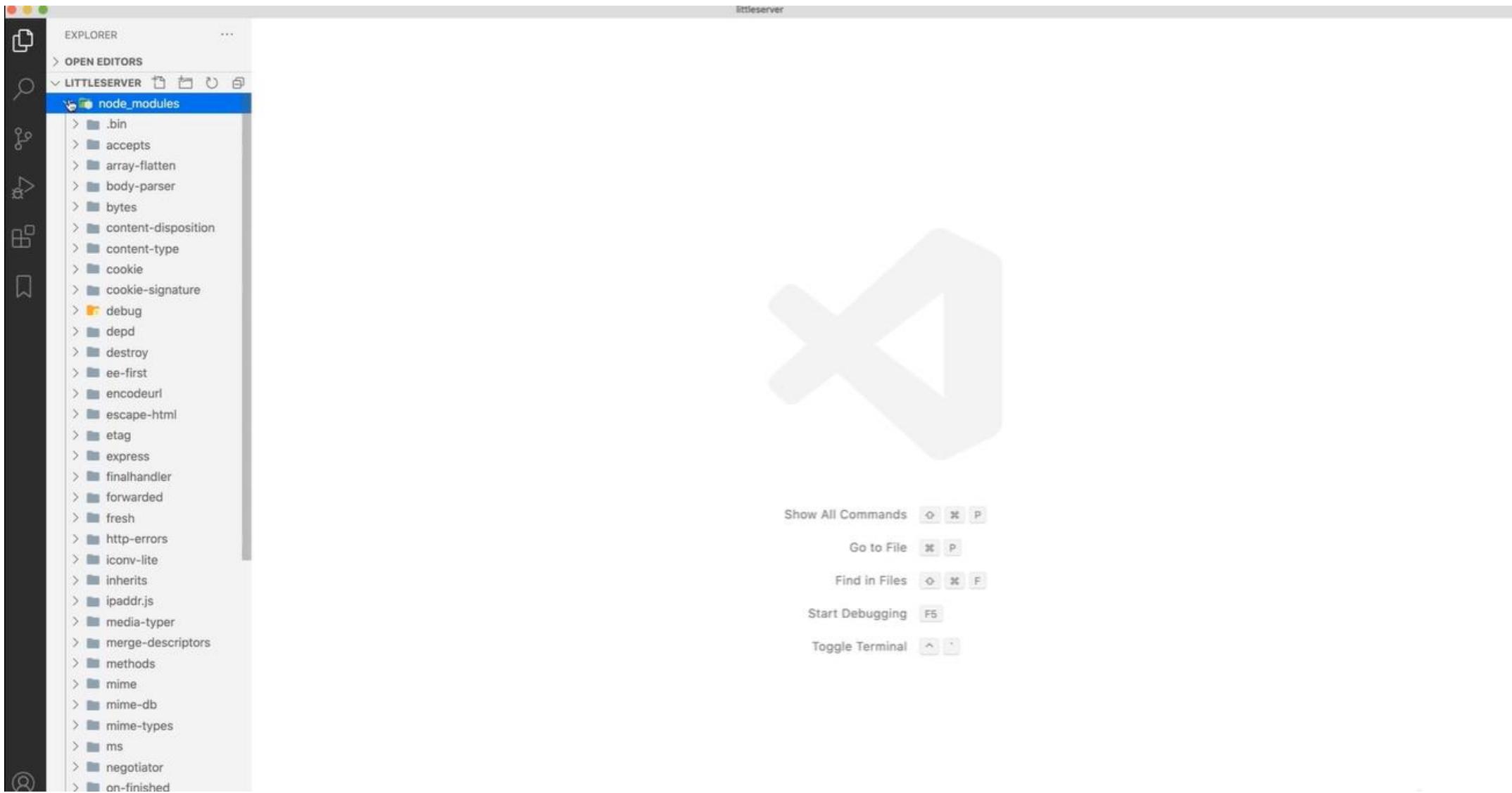


```
"version": "1.0.0",
"description": "small server",
"main": "index.js",
"scripts": {
  "test": "echo \\\"Error: no test specified\\\" && exit 1"
},
"author": "abel@mit.edu",
"license": "MIT",
"dependencies": {
  "express": "^4.17.1"
}
✓ littleserver %
```

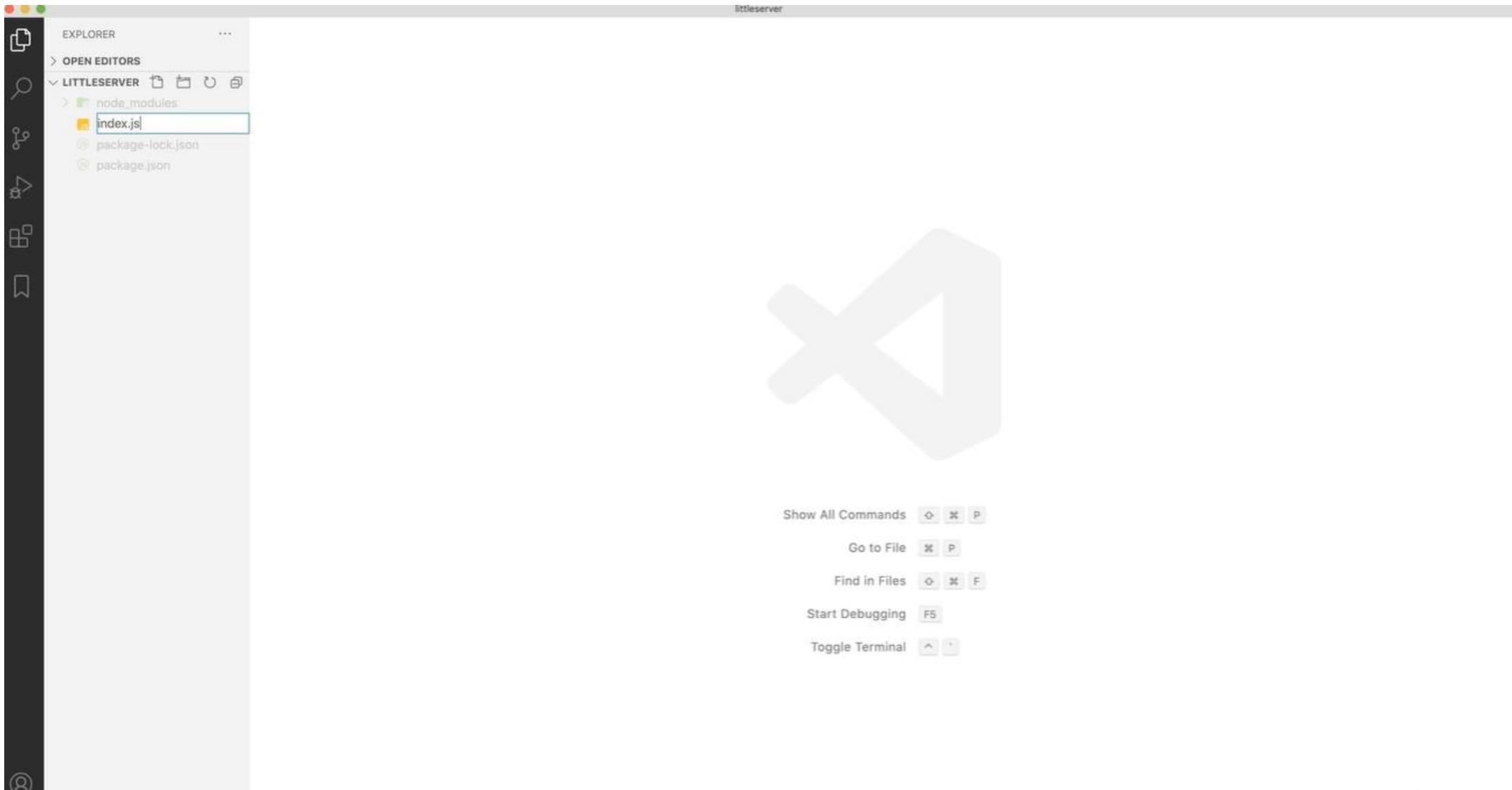
# Creating Little Server (13/20)



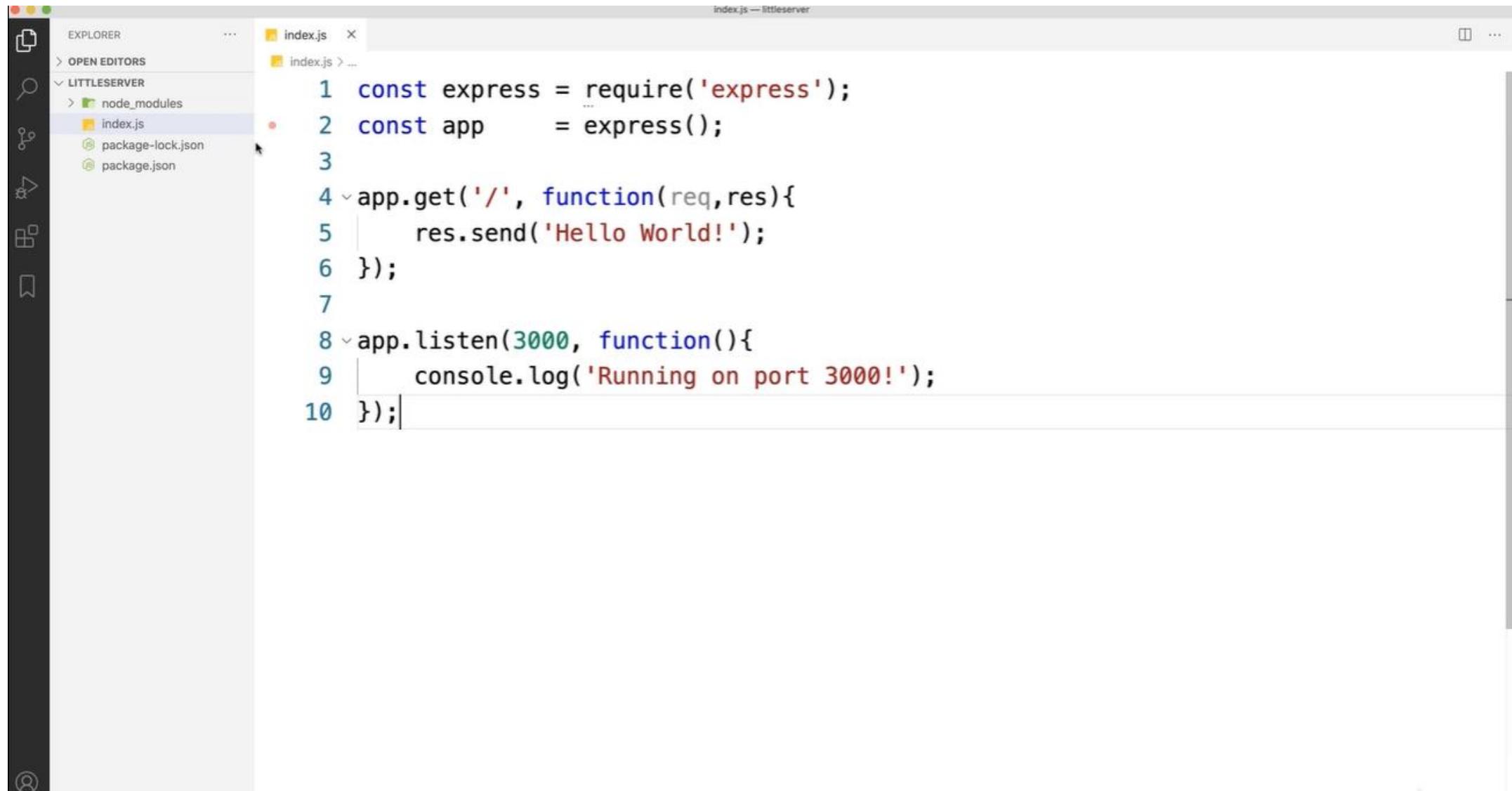
# Creating Little Server (14/20)



# Creating Little Server (15/20)



# Creating Little Server (16/20)

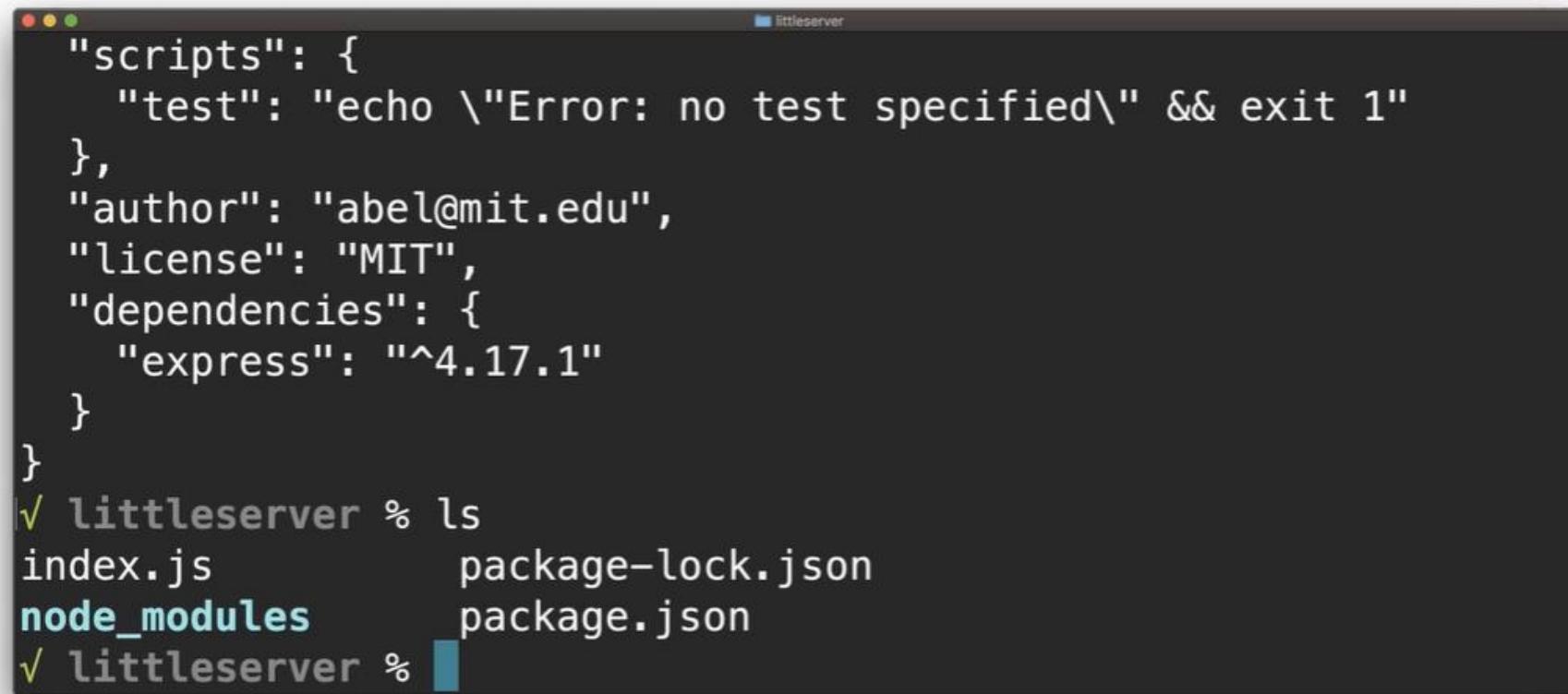


The screenshot shows a code editor window with the following details:

- Title Bar:** index.js — littleserver
- Explorer Bar (Left):** Shows a project structure under "LITTLE SERVER": node\_modules, index.js (selected), package-lock.json, and package.json.
- Editor Area (Right):** Displays the contents of index.js with line numbers 1 through 10. The code uses ES6 syntax to import express, create an app, define a route to return "Hello World!", and start the server on port 3000.

```
index.js
1 const express = require('express');
2 const app = express();
3
4 app.get('/', function(req, res){
5   res.send('Hello World!');
6 });
7
8 app.listen(3000, function(){
9   console.log('Running on port 3000!');
10});
```

# Creating Little Server (17/20)



The terminal window shows the creation of a new Node.js project named 'littleserver'. It displays the contents of the package.json file and the results of an 'ls' command.

```
"scripts": {  
  "test": "echo \\\"Error: no test specified\\\" && exit 1"  
},  
"author": "abel@mit.edu",  
"license": "MIT",  
"dependencies": {  
  "express": "^4.17.1"  
}  
}  
✓ littleserver % ls  
index.js          package-lock.json  
node_modules    package.json  
✓ littleserver %
```

## Creating Little Server (18/20)



```
littleserver % node index.js
Running on port 3000!
```

# Creating Little Server (19/20)

The screenshot shows the npmjs.com page for the 'express' package. At the top, there's a browser-like header with tabs for 'localhost:3000' and a search bar for 'localhost:3000 - Google Search'. Below the header, the package name 'express' is displayed in large, lowercase letters. To the right of the name are navigation links: 'Readme' (highlighted in yellow), 'Explore (BETA)', '30 Dependencies', '46,655 Dependents', and '264 Versions'. On the left side of the main content area, there's a snippet of Node.js code:

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

app.get('/', function (req, res) {
  res.send('Hello World')
})

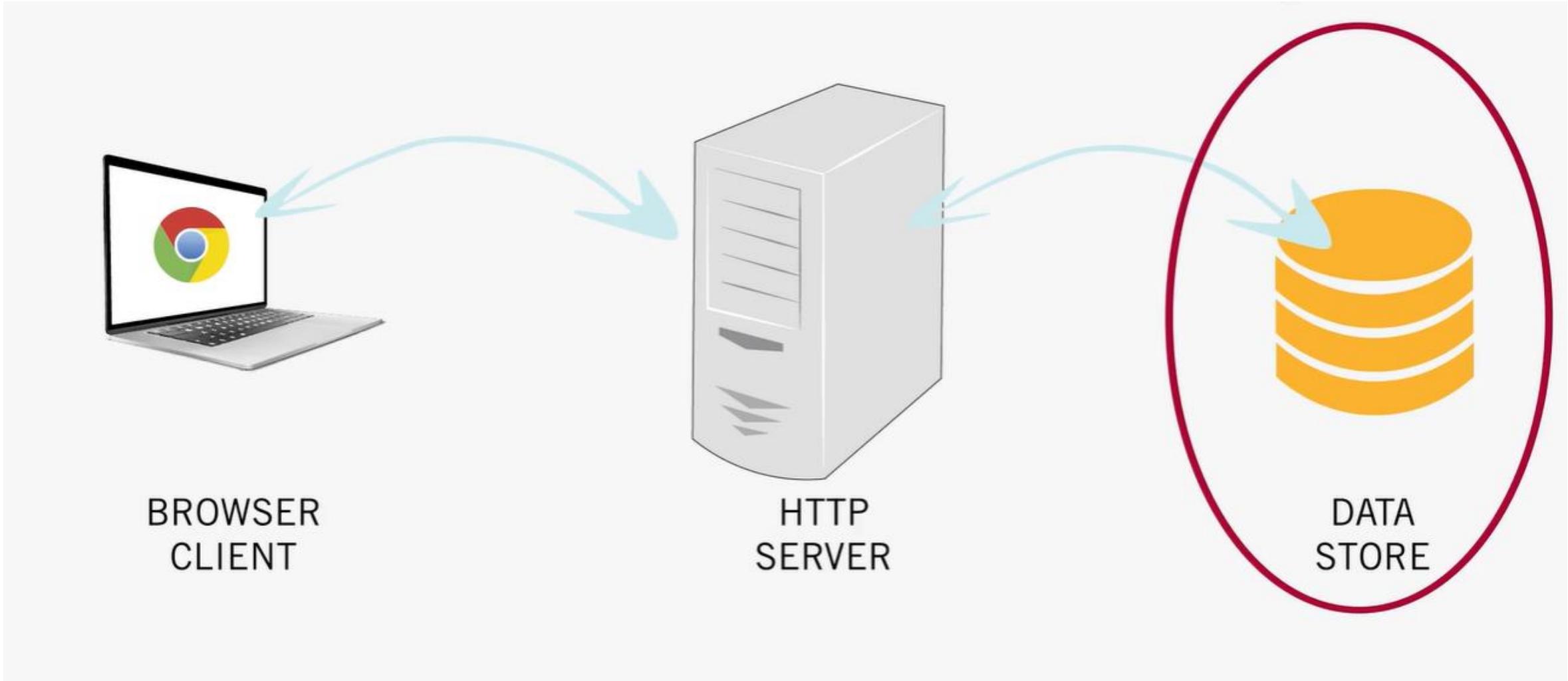
app.listen(3000)
```

Below the code, there are badge indicators for npm version (v4.17.1), downloads (59M/month), Linux (passing), Windows (passing), and coverage (100%). To the right of the code, there's an 'Install' section with the command 'npm i express'. Further down, there's a chart showing weekly downloads at 14,057,862, followed by version information (4.17.1), license (MIT), unpacked size (208 kB), total files (16), issues (97), pull requests (52), and a link to the homepage (expressjs.com/).

# Creating Little Server (20/20)



# Three Tiers: Client, Server, & Storage



## Low DB



```
[ { id: 4, title: 'great', published: 'true' } ]
Done reading data store!
abel@mbp:~/dropbox/courses/100/github/one_2019/badbank/02_littleDB$ node simple_
read.js
[ { id: 1, title: 'lowdb is awesome', published: 'false' },
  { id: 2, title: 'good', published: 'false' },
  { id: 3, title: 'cool', published: 'true' },
  { id: 4, title: 'great', published: 'true' } ]
Done reading data store!
abel@mbp:~/dropbox/courses/100/github/one_2019/badbank/02_littleDB$ node simple_
write.js
Done writing to data store!
abel@mbp:~/dropbox/courses/100/github/one_2019/badbank/02_littleDB$ node simple_
read.js
[ { id: 1, title: 'lowdb is awesome', published: 'false' },
  { id: 2, title: 'good', published: 'false' },
  { id: 3, title: 'cool', published: 'true' },
  { id: 4, title: 'great', published: 'true' },
  { id: 5, title: 'boston', published: 'false' },
  { id: 6, title: 'NYC', published: 'true' },
  { id: 7, title: 'DC', published: 'true' } ]
Done reading data store!
abel@mbp:~/dropbox/courses/100/github/one_2019/badbank/02_littleDB$ node simple_
```

# Creating Little DB (1/22)

lowdb - npm Not Secure | npmjs.com/package/lowdb

**lowdb**  
1.0.0 • Public • Published 3 years ago

[Readme](#) [Explore BETA](#) [5 Dependencies](#) [778 Dependents](#) [56 Versions](#)

## Lowdb

downloads 1.1M/month npm package 1.0.0 build passing patreon donate

Small JSON database for Node, Electron and the browser. Powered by Lodash. ⚡

```
db.get('posts')
  .push({ id: 1, title: 'lowdb is awesome' })
  .write()
```

## Usage

```
npm install lowdb
```

```
const low = require('lowdb')
```

Install

```
> npm i lowdb
```

Weekly Downloads

262,956

Version	1.0.0
License	MIT
Issues	94
Pull Requests	12

Homepage  
[github.com/typicode/lowdb](https://github.com/typicode/lowdb)

Repository

# Creating Little DB (2/22)

lowdb - npm Not Secure | npmjs.com/package/lowdb

## Usage

```
npm install lowdb
```

```
const low = require('lowdb')
const FileSync = require('lowdb/adapters/FileSync')

const adapter = new FileSync('db.json')
const db = low(adapter)

// Set some defaults
db.defaults({ posts: [], user: {} })
  .write()

// Add a post
db.get('posts')
  .push({ id: 1, title: 'lowdb is awesome' })
  .write()

// Set a user using Lodash shorthand syntax
db.set('user.name', 'typicode')
  .write()
```

94 12

Homepage [github.com/typicode/lowdb](https://github.com/typicode/lowdb)

Repository [github.com/typicode/lowdb](https://github.com/typicode/lowdb)

Last publish 3 years ago

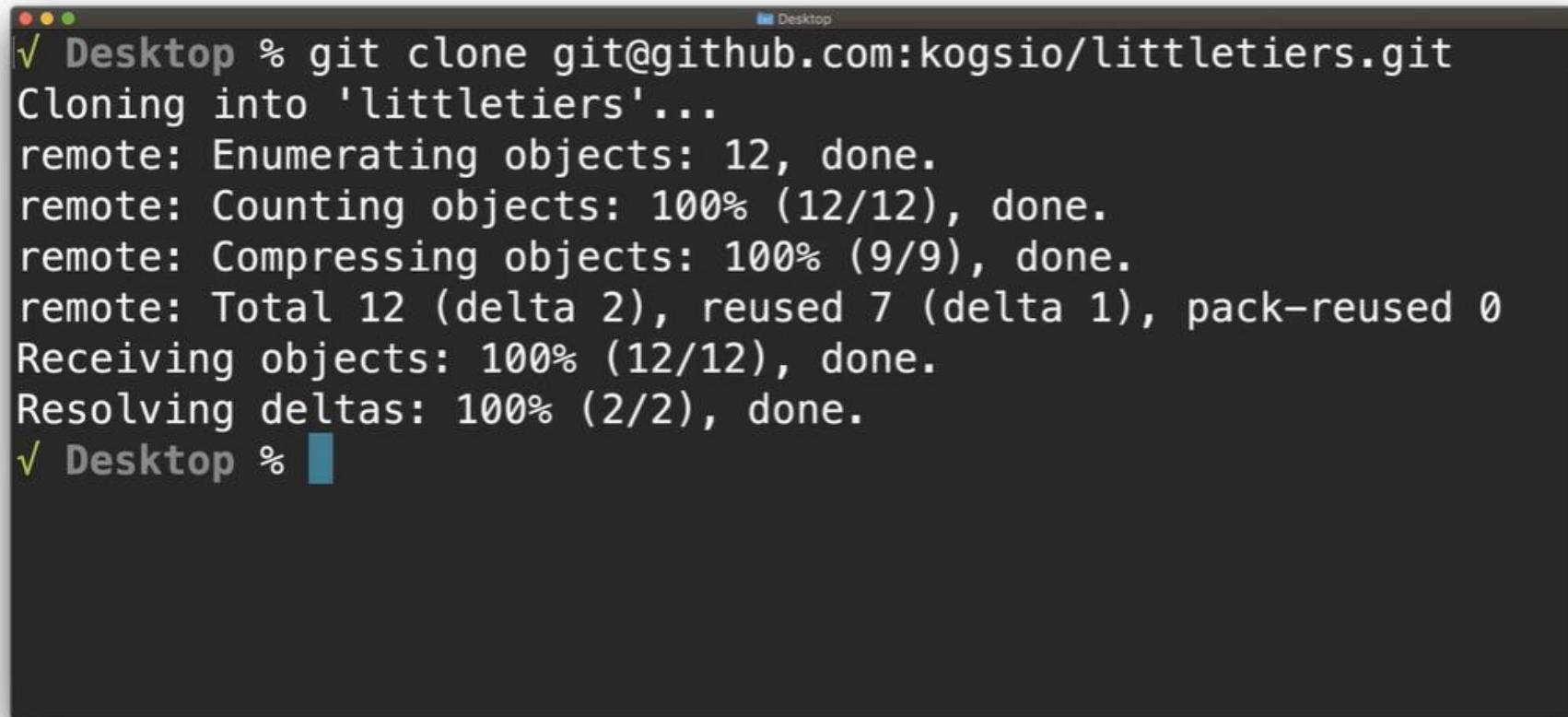
Collaborators



> Try on RunKit

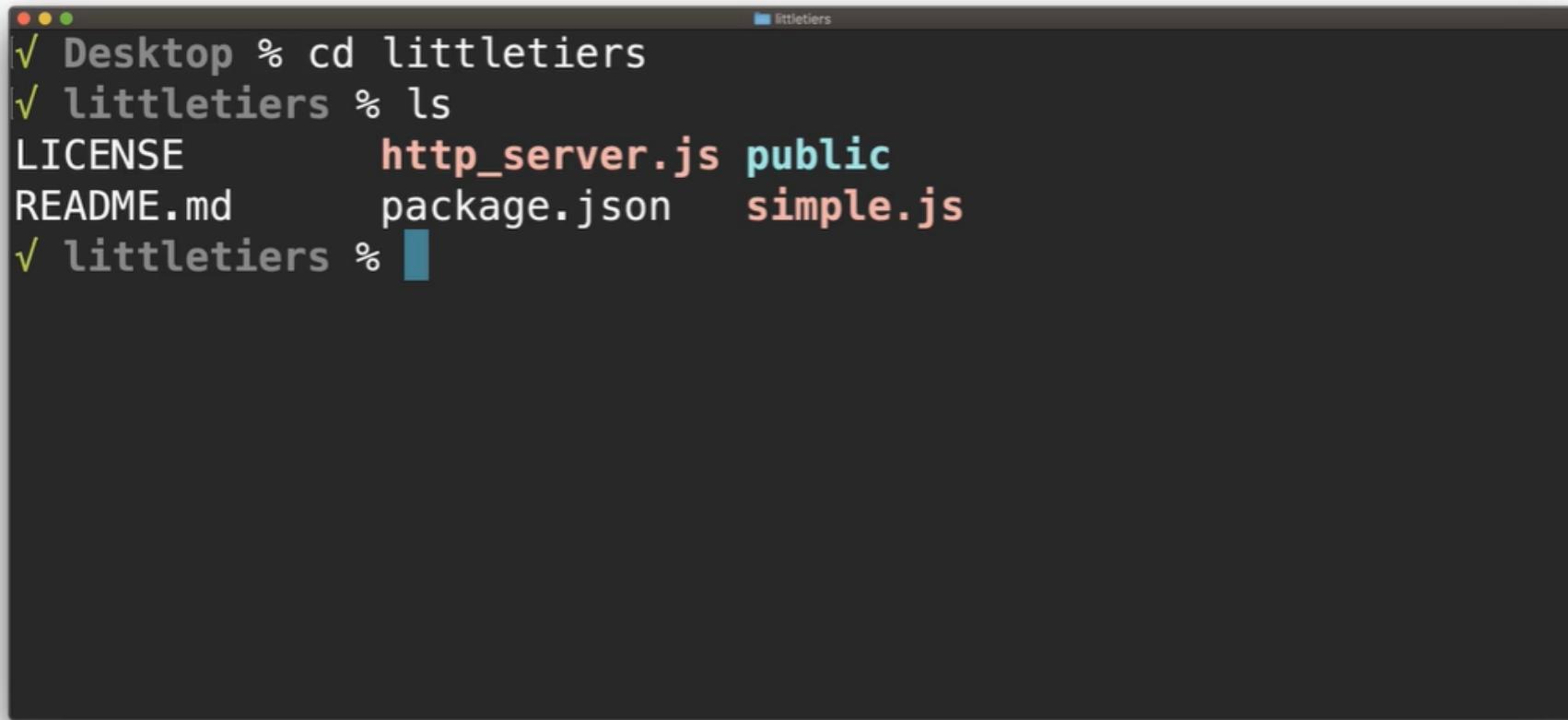
Report a vulnerability

## Creating Little DB (3/22)



```
Desktop % git clone git@github.com:kogsio/littletiers.git
Cloning into 'littletiers'...
remote: Enumerating objects: 12, done.
remote: Counting objects: 100% (12/12), done.
remote: Compressing objects: 100% (9/9), done.
remote: Total 12 (delta 2), reused 7 (delta 1), pack-reused 0
Receiving objects: 100% (12/12), done.
Resolving deltas: 100% (2/2), done.
Desktop %
```

## Creating Little DB (4/22)

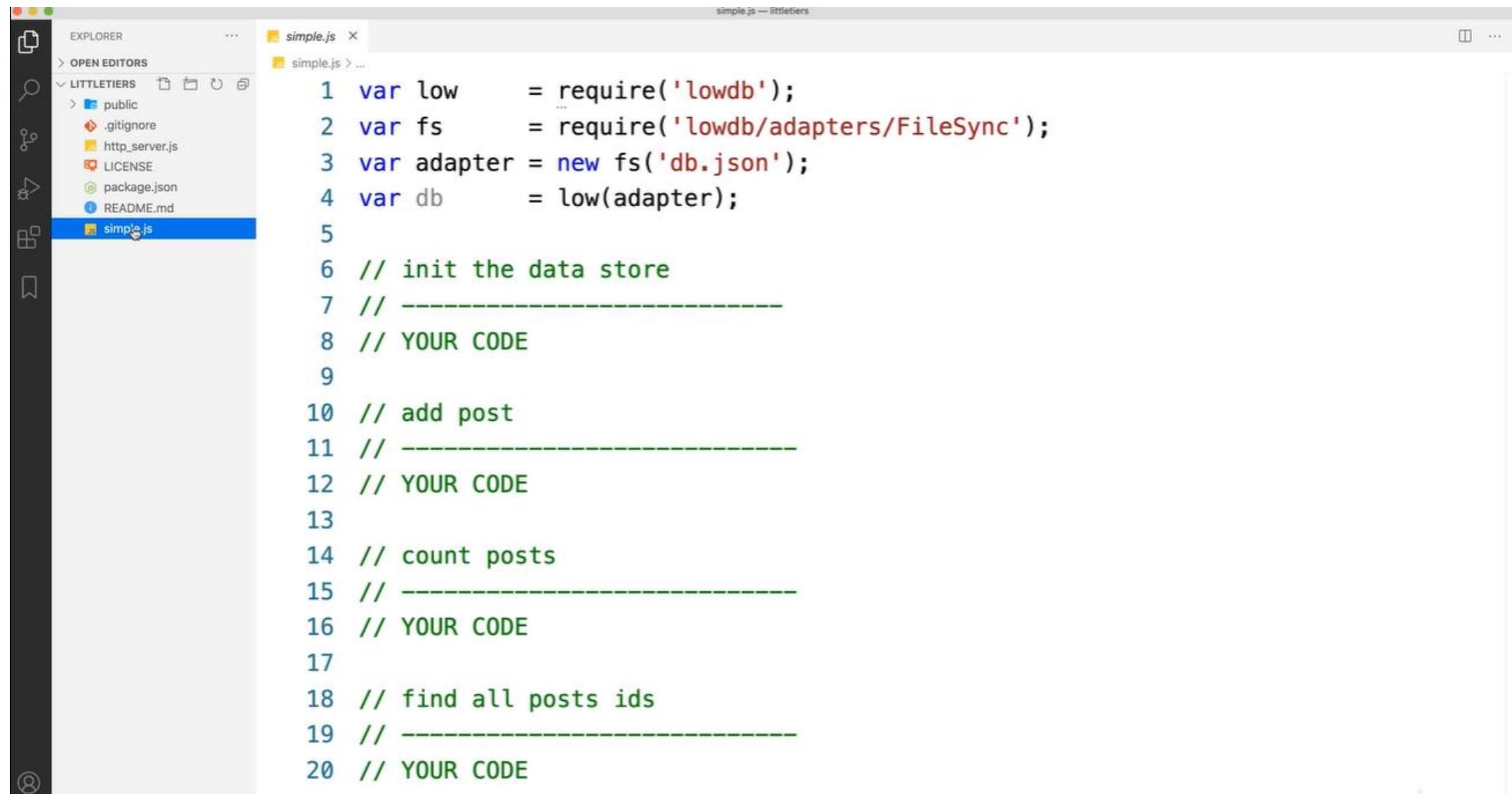


A screenshot of a macOS terminal window titled "littletiers". The window shows the following command-line session:

```
Desktop % cd littletiers
littletiers % ls
LICENSE          http_server.js  public
README.md        package.json   simple.js
littletiers %
```

The terminal window has a dark background and light-colored text. The cursor is positioned at the end of the final command line.

# Creating Little DB (5/22)



The screenshot shows a code editor interface with a dark theme. On the left is the Explorer sidebar, which lists a project structure under 'LITTLETIERS'. The 'simple.js' file is selected and highlighted with a blue bar at the bottom of the list. The main editor area shows the following code:

```
simple.js — littletiers
simple.js > ...
1 var low      = require('lowdb');
2 var fs       = require('lowdb/adapters/FileSync');
3 var adapter = new fs('db.json');
4 var db       = low(adapter);
5
6 // init the data store
7 // -----
8 // YOUR CODE
9
10 // add post
11 // -----
12 // YOUR CODE
13
14 // count posts
15 // -----
16 // YOUR CODE
17
18 // find all posts ids
19 // -----
20 // YOUR CODE
```

# Creating Little DB (6/22)

lowdb - npm

Not Secure | npmjs.com/package/lowdb

Usage

```
npm install lowdb
```

```
const low = require('lowdb')
const FileSync = require('lowdb/adapters/FileSync')

const adapter = new FileSync('db.json')
const db = low(adapter)

// Set some defaults
db.defaults({ posts: [], user: {} })
  .write()

// Add a post
db.get('posts')
  .push({ id: 1, title: 'lowdb is awesome' })
  .write()

// Set a user using Lodash shorthand syntax
db.set('user.name', 'typicode')
  .write()
```

94      12

Homepage [github.com/typicode/lowdb](https://github.com/typicode/lowdb)

Repository [github.com/typicode/lowdb](https://github.com/typicode/lowdb)

Last publish 3 years ago

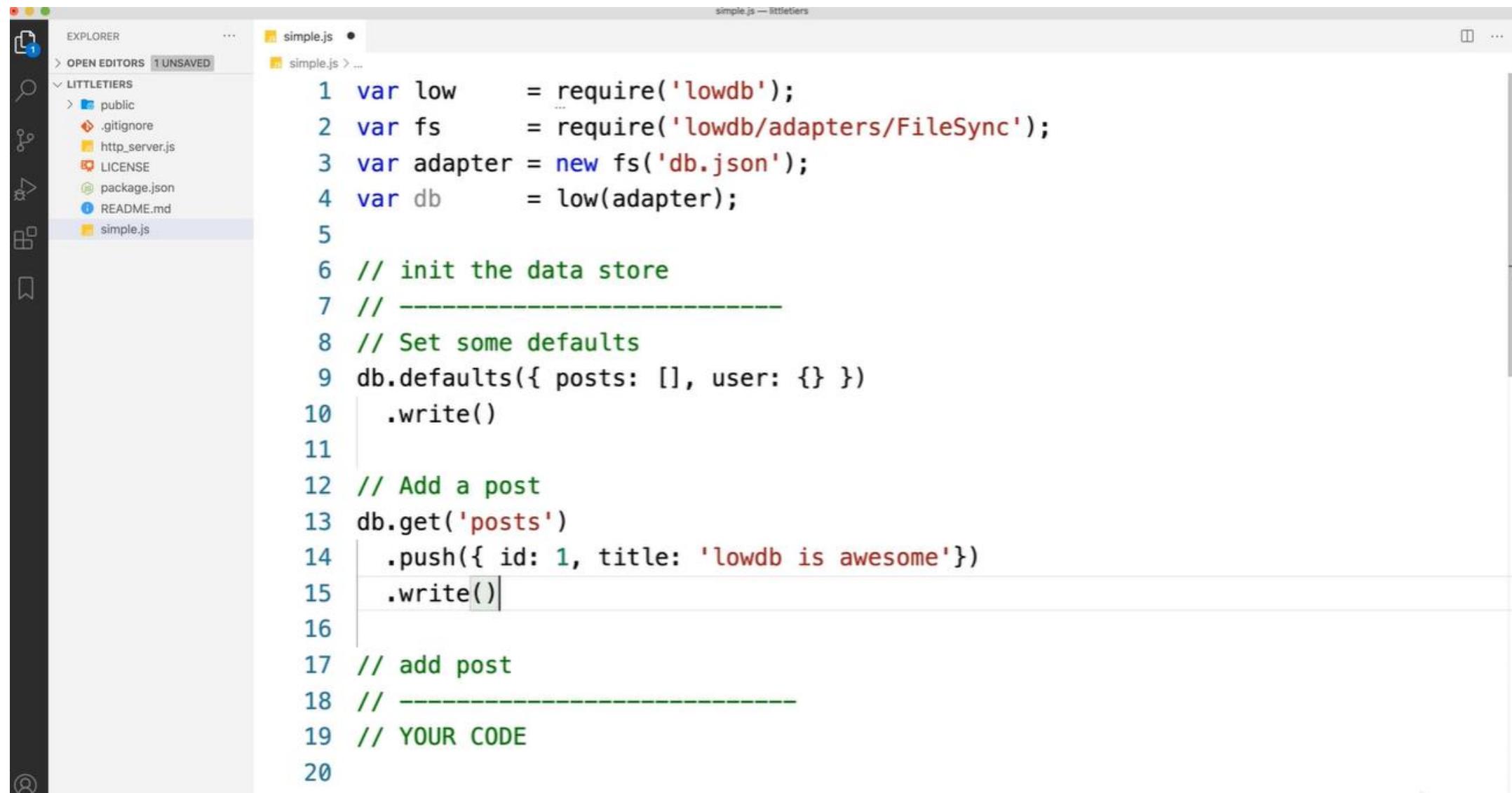
Collaborators



> Try on RunKit

Report a vulnerability

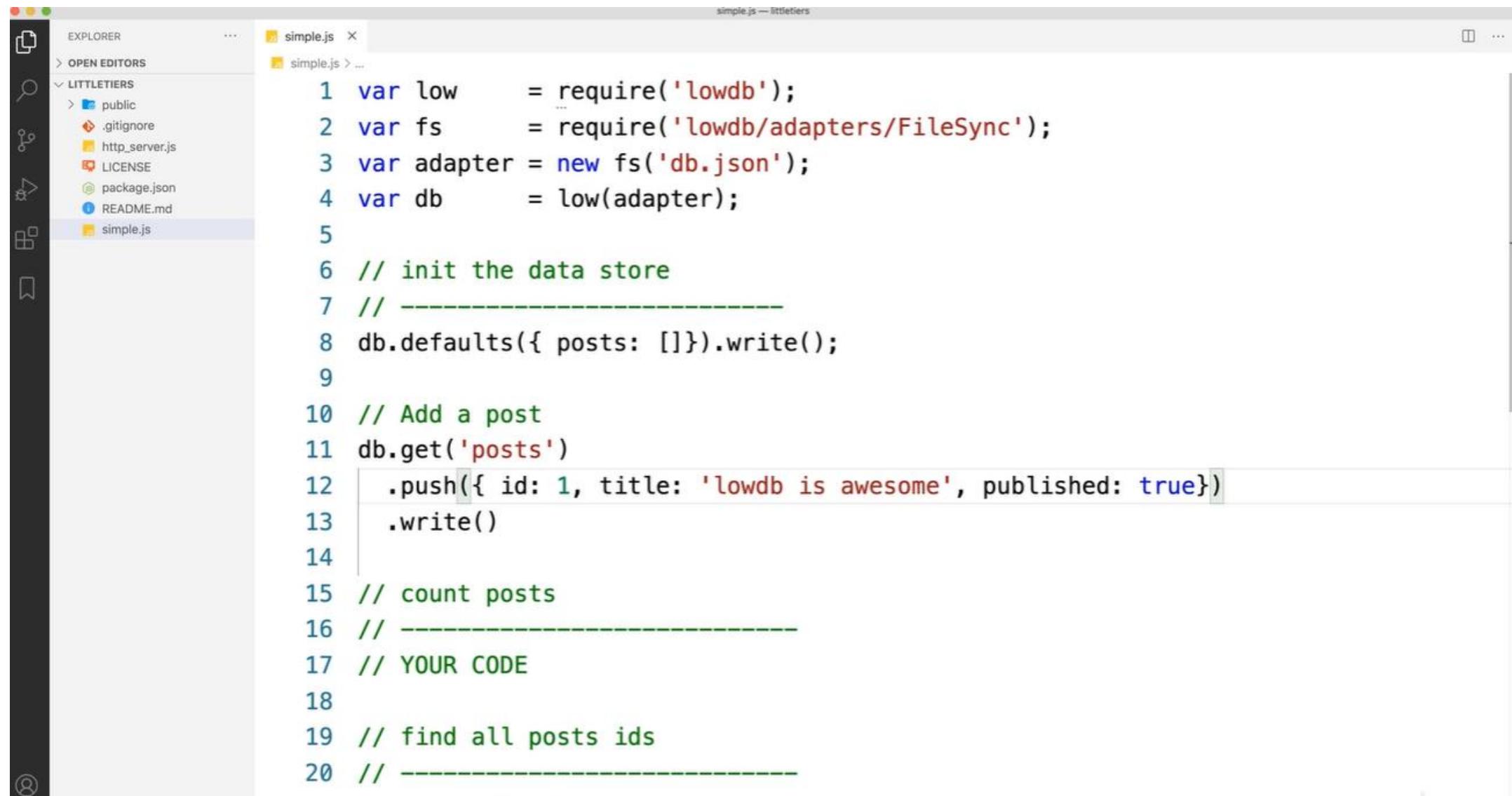
# Creating Little DB (7/22)



The screenshot shows a Microsoft Visual Studio Code (VS Code) interface. On the left is the Explorer sidebar, which lists the project structure under 'LITTLETIERS'. The 'simple.js' file is selected in the list. The main editor area displays the code for 'simple.js'.

```
simple.js • simple.js > ...
1 var low      = require('lowdb');
2 var fs       = require('lowdb/adapters/FileSync');
3 var adapter = new fs('db.json');
4 var db       = low(adapter);
5
6 // init the data store
7 // -----
8 // Set some defaults
9 db.defaults({ posts: [], user: {} })
10 .write()
11
12 // Add a post
13 db.get('posts')
14 .push({ id: 1, title: 'lowdb is awesome'})
15 .write()
16
17 // add post
18 // -----
19 // YOUR CODE
20
```

# Creating Little DB (8/22)



The screenshot shows a code editor interface with a dark theme. On the left is the Explorer sidebar, which lists a project structure under 'LITTLETIERS': public, .gitignore, http\_server.js, LICENSE, package.json, README.md, and simple.js. The simple.js file is currently selected and open in the main editor area. The code itself is as follows:

```
simple.js — littletiers
simple.js > ...
1 var low      = require('lowdb');
2 var fs       = require('lowdb/adapters/FileSync');
3 var adapter = new fs('db.json');
4 var db       = low(adapter);
5
6 // init the data store
7 // -----
8 db.defaults({ posts: [] }).write();
9
10 // Add a post
11 db.get('posts')
12   .push({ id: 1, title: 'lowdb is awesome', published: true })
13   .write()
14
15 // count posts
16 // -----
17 // YOUR CODE
18
19 // find all posts ids
20 // -----
```

# Creating Little DB (9/22)

The screenshot shows a terminal window within a code editor interface. The terminal output is as follows:

```
Desktop % cd littletiers
littletiers % ls
LICENSE      http_server.js  public
README.md    package.json    simple.js
littletiers % node simple.js
```

The code editor shows a file named `simple.js` with the following content:

```
1 var low      = require('lowdb');
2 var fs        = require('lowdb/adapters/FileSync');

// YOUR CODE
18
19 // find all posts ids
20 // -----
```

# Creating Little DB (10/22)

The screenshot shows a Visual Studio Code interface. On the left is the Explorer sidebar with a tree view of a project named 'LITTLETIERS'. The 'simple.js' file is open in the main editor area. A terminal window is also present, displaying a stack trace and a command prompt.

```
simple.js — littletiers
simple.js
1 var low      = require('lowdb');
2 var fs        = require('lowdb/adapters/FileSync');

e.js:1:15)
    at Module._compile (internal/modules/cjs/loader.js:1137:30)
    at Object.Module._extensions..js (internal/modules/cjs/loader.js:1157:10)
    at Module.load (internal/modules/cjs/loader.js:985:32)
    at Function.Module._load (internal/modules/cjs/loader.js:878:14)
    at Function.executeUserEntryPoint [as runMain] (internal/modules/run_main.js:71:12) {
  code: 'MODULE_NOT_FOUND',
  requireStack: [ '/Users/abel/Desktop/littletiers/simple.js' ]
}
?1 littletiers % npm instal lowdb
17 // YOUR CODE
18
19 // find all posts ids
20 // -----
```

# Creating Little DB (11/22)

The screenshot shows a terminal window in a code editor interface. The terminal is running on a Mac OS X system, as indicated by the window title bar and the dock icon.

The terminal output is as follows:

```
simple.js — littletiers
simple.js:1:15)
    at Module._compile (internal/modules/cjs/loader.js:1137:30)
    at Object.Module._extensions..js (internal/modules/cjs/loader.js:1157:10)
    at Module.load (internal/modules/cjs/loader.js:985:32)
    at Function.Module._load (internal/modules/cjs/loader.js:878:14)
    at Function.executeUserEntryPoint [as runMain] (internal/modules/run_main.js:71:12) {
  code: 'MODULE_NOT_FOUND',
  requireStack: [ '/Users/abel/Desktop/littletiers/simple.js' ]
}
?1 littletiers % cat package.json
```

The code editor's sidebar shows a project structure with files like `simple.js`, `public`, `.gitignore`, `http`, `LIC`, `package.json`, `README.md`, and `simple.js`.

# Creating Little DB (12/22)

The screenshot shows a terminal window within a code editor interface. The terminal is running on a Mac OS X system, indicated by the window title bar and the Dock icon at the bottom.

The terminal command being run is:

```
littletiers % npm install lowdb
```

The output of the command is displayed in the terminal window:

```
« [REDACTED] » :: extract:lowdb: sill extract lowdb@^1.0.0
```

Below the terminal window, the code editor displays a file named `simple.js` with the following content:

```
1 var low      = require('lowdb');
2 var fs       = require('lowdb/adapters/FileSync');

"author": "abel@mit.edu",
"license": "MIT",
"bugs": {
  "url": "https://github.com/kogsio/littletiers/issues"
},
"homepage": "https://github.com/kogsio/littletiers#readme",
"dependencies": {
  "express": "^4.17.1",
  "lowdb": "^1.0.0"
}
littletiers % npm install lowdb
17 // YOUR CODE
18
19 // find all posts ids
20 // -----
```

The code editor's sidebar shows a project structure under `LITTLETIERS`, including files like `.gitignore`, `http`, `LIC`, `pac`, `RE`, and `sim`.

# Creating Little DB (13/22)

The screenshot shows a Visual Studio Code interface. On the left is the Explorer sidebar with a tree view of the project structure, including 'LITTLETIERS' (with 'node\_modules' and 'public' folders), '.git', and several package.json files. The main area has two tabs: 'simple.js' and 'simple.js > ...'. The code editor shows the following content:

```
simple.js — littletiers
simple.js
1 var low      = require('lowdb');
2 var fs       = require('lowdb/adapters/FileSync');

npm notice created a lockfile as package-lock.json. You should c
ommit this file.

+ lowdb@1.0.0
added 6 packages from 6 contributors and audited 6 packages in 0
.488s
found 0 vulnerabilities

✓ littletiers % npm install
added 50 packages from 37 contributors and audited 56 packages i
n 1.129s
found 0 vulnerabilities

✓ littletiers %
17 // ----
18
19 // find all posts ids
20 // -----
```

The terminal window below the code editor displays the output of the 'npm install' command, showing the addition of 50 packages from 37 contributors and 56 audited packages in 1.129s, with 0 vulnerabilities found.

# Creating Little DB (14/22)

The screenshot shows a terminal window within a code editor interface. The terminal output is as follows:

```
simple.js — littletiers
simple.js
1 var low      = require('lowdb');
2 var fs       = require('lowdb/adapters/FileSync'):
n 1.129s
found 0 vulnerabilities

✓ littletiers % ls
LICENSE          node_modules      public
README.md        package-lock.json simple.js
http_server.js   package.json

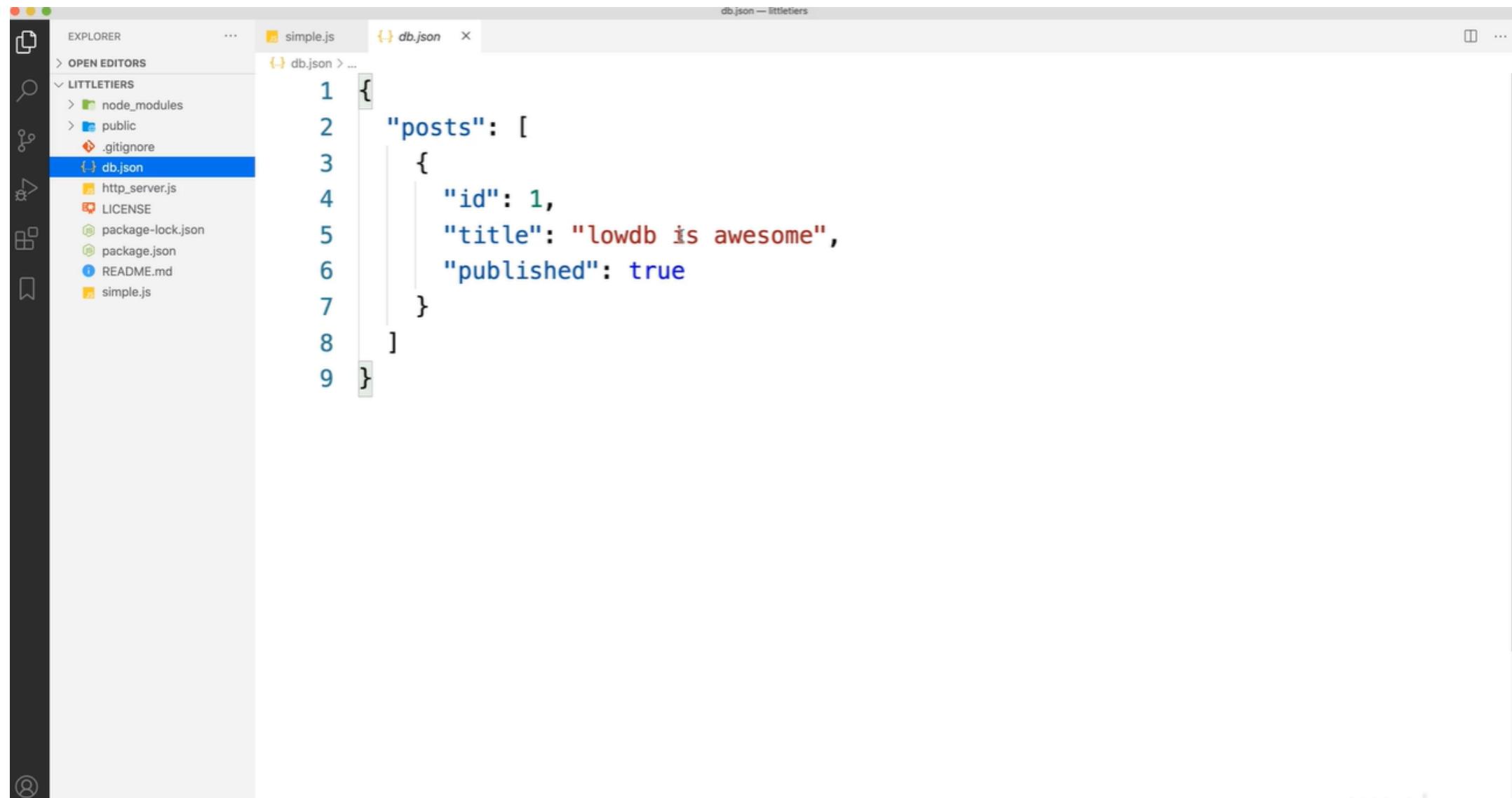
✓ littletiers % node simple.js

✓ littletiers % ls
LICENSE          http_server.js   package.json
README.md        node_modules      public
db.json          package-lock.json simple.js

✓ littletiers %
```

The terminal shows the execution of a Node.js script named `simple.js` which uses the `lowdb` library to create a database. It then lists the contents of the current directory, which includes files like `LICENSE`, `README.md`, `http_server.js`, `node_modules`, `public`, `simple.js`, and `db.json`. Finally, it runs the `simple.js` script again and lists the directory contents again, showing that the database file `db.json` has been created.

# Creating Little DB (15/22)



The screenshot shows a code editor interface with the following details:

- Explorer Bar:** On the left, it shows a tree view of the project structure under "LITTLETIERS". The files listed are node\_modules, public, .gitignore, db.json (which is selected and highlighted in blue), http\_server.js, LICENSE, package-lock.json, package.json, README.md, and simple.js.
- Editor Area:** The main area displays the contents of db.json. The JSON structure is as follows:

```
1 {  
2   "posts": [  
3     {  
4       "id": 1,  
5       "title": "lowdb is awesome",  
6       "published": true  
7     }  
8   ]  
9 }
```

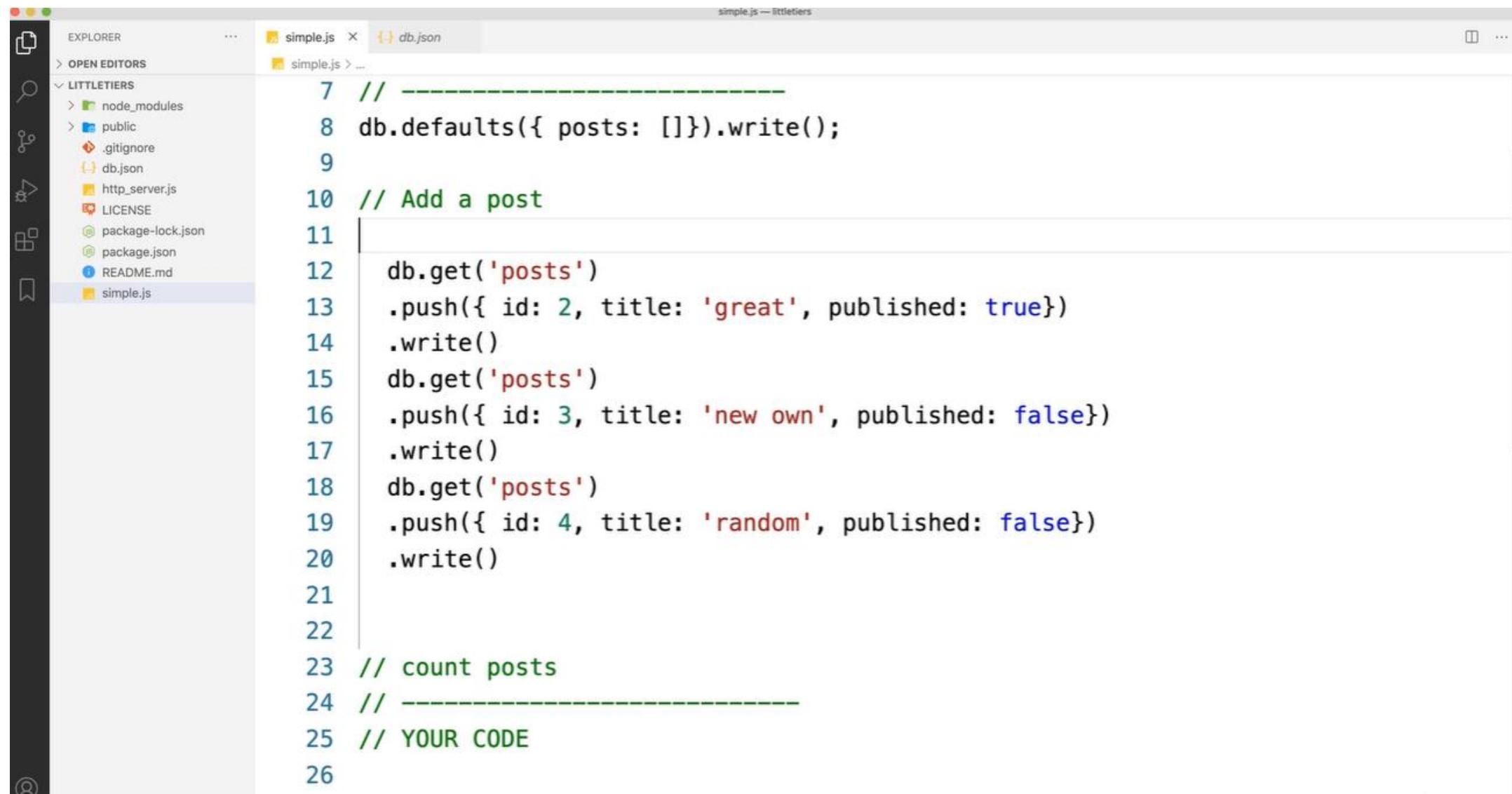
# Creating Little DB (16/22)

The screenshot shows a code editor interface with a dark theme. On the left is the Explorer sidebar, which lists the project structure under 'LITTLETIERS'. The 'simple.js' file is selected in the sidebar. The main area displays the contents of 'simple.js'.

```
simple.js — littletiers
simple.js x db.json
simple.js > published

7 // -----
8 db.defaults({ posts: [] }).write();
9
10 // Add a post
11 db.get('posts')
12   .push({ id: 1, title: 'lowdb is awesome', published: true })
13   .write()
14
15 db.get('posts')
16   .push({ id: 2, title: 'great', published: true })
17   .write()
18 db.get('posts')
19   .push({ id: 3, title: 'new own', published: false })
20   .write()
21 db.get('posts')
22   .push({ id: 4, title: 'random', published: false })
23   .write()
24
25
26 // count posts
```

# Creating Little DB (17/22)



The screenshot shows a Microsoft Visual Studio Code (VS Code) interface. On the left is the Explorer sidebar, which lists the project structure under 'LITTLETIERS'. The 'simple.js' file is currently open in the editor. The code in 'simple.js' is as follows:

```
simple.js — littletiers
simple.js x db.json
simple.js > ...

7 // -----
8 db.defaults({ posts: [] }).write();
9
10 // Add a post
11
12 db.get('posts')
13 .push({ id: 2, title: 'great', published: true })
14 .write()
15 db.get('posts')
16 .push({ id: 3, title: 'new own', published: false })
17 .write()
18 db.get('posts')
19 .push({ id: 4, title: 'random', published: false })
20 .write()
21
22
23 // count posts
24 // -----
25 // YOUR CODE
26
```

# Creating Little DB (18/22)

The screenshot shows a terminal window within a code editor interface. The terminal output is as follows:

```
✓ littletiers % ls
LICENSE          node_modules      public
README.md        package-lock.json simple.js
http_server.js   package.json

✓ littletiers % node simple.js
✓ littletiers % ls
LICENSE          http_server.js  package.json
README.md        node_modules    public
db.json          package-lock.json simple.js
✓ littletiers % node simple.js
✓ littletiers %
```

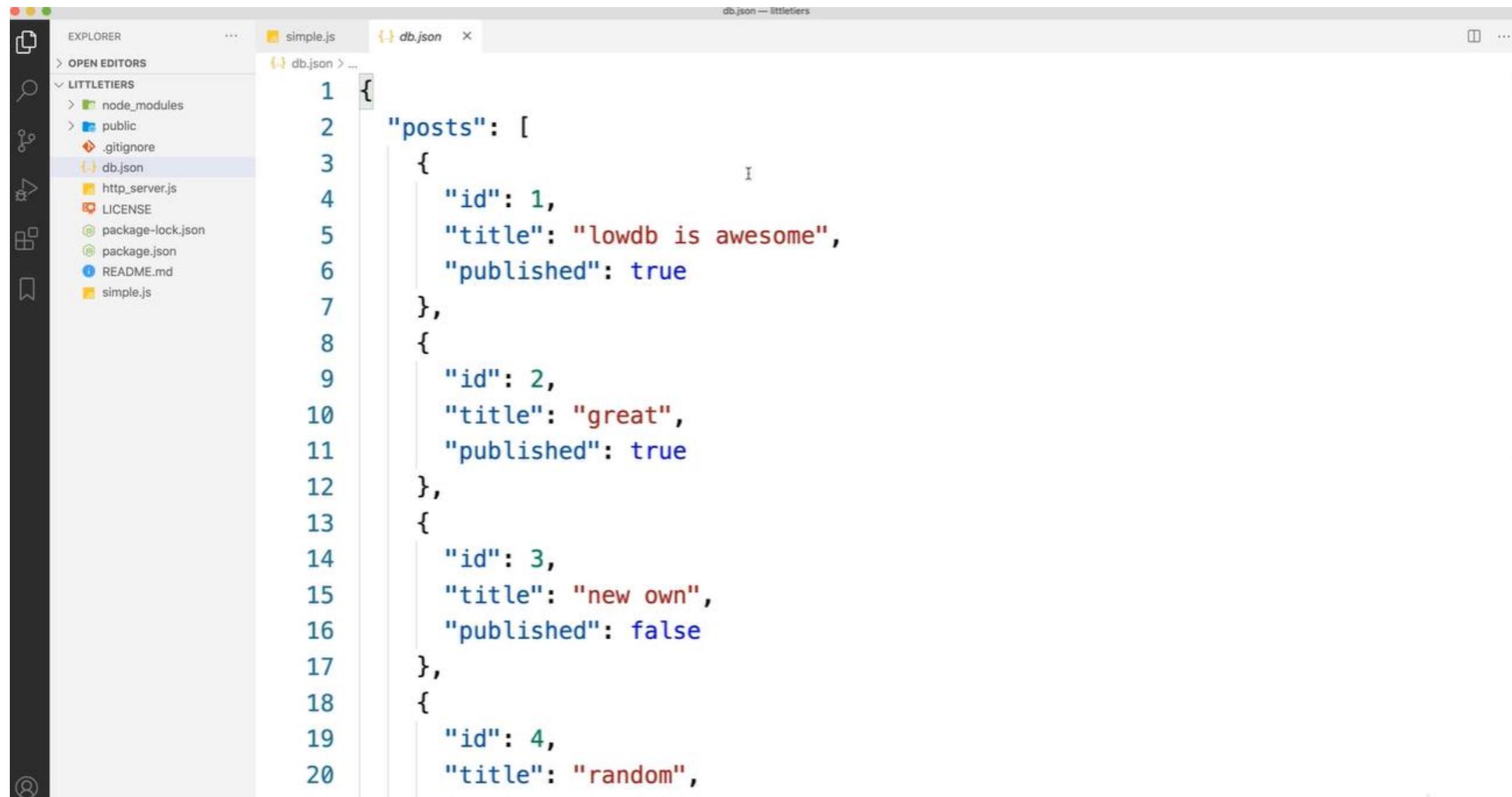
At the bottom of the terminal, there is some partially visible code:

```
23 // count posts
24 // -----
25 // YOUR CODE
26
```

The code editor's sidebar shows a project structure with files like `simple.js`, `db.json`, and `http_server.js`. The `simple.js` file contains the following code:

```
7 // -----
8 db.defaults({ posts: [] }).write();
found 0 vulnerabilities
```

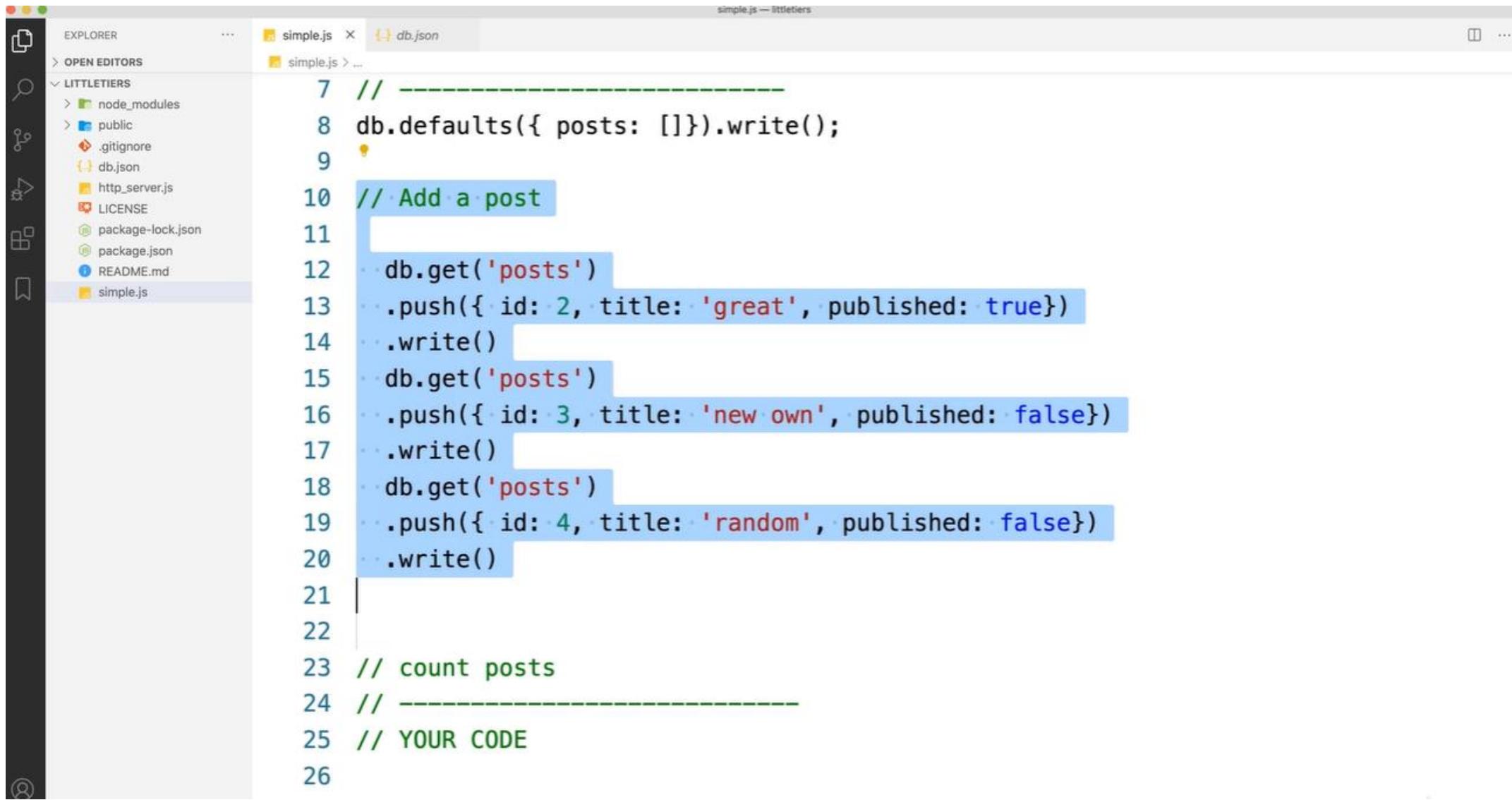
# Creating Little DB (19/22)



The screenshot shows a Microsoft Visual Studio Code (VS Code) interface. On the left is the Explorer sidebar, which lists the project structure under 'LITTLETIERS'. The 'db.json' file is selected in the list. The main editor area displays the JSON content of 'db.json'.

```
1 {  
2   "posts": [  
3     {  
4       "id": 1,  
5       "title": "lowdb is awesome",  
6       "published": true  
7     },  
8     {  
9       "id": 2,  
10      "title": "great",  
11      "published": true  
12    },  
13    {  
14      "id": 3,  
15      "title": "new own",  
16      "published": false  
17    },  
18    {  
19      "id": 4,  
20      "title": "random",  
21    }  
22  ]  
23}  
24}
```

# Creating Little DB (20/22)

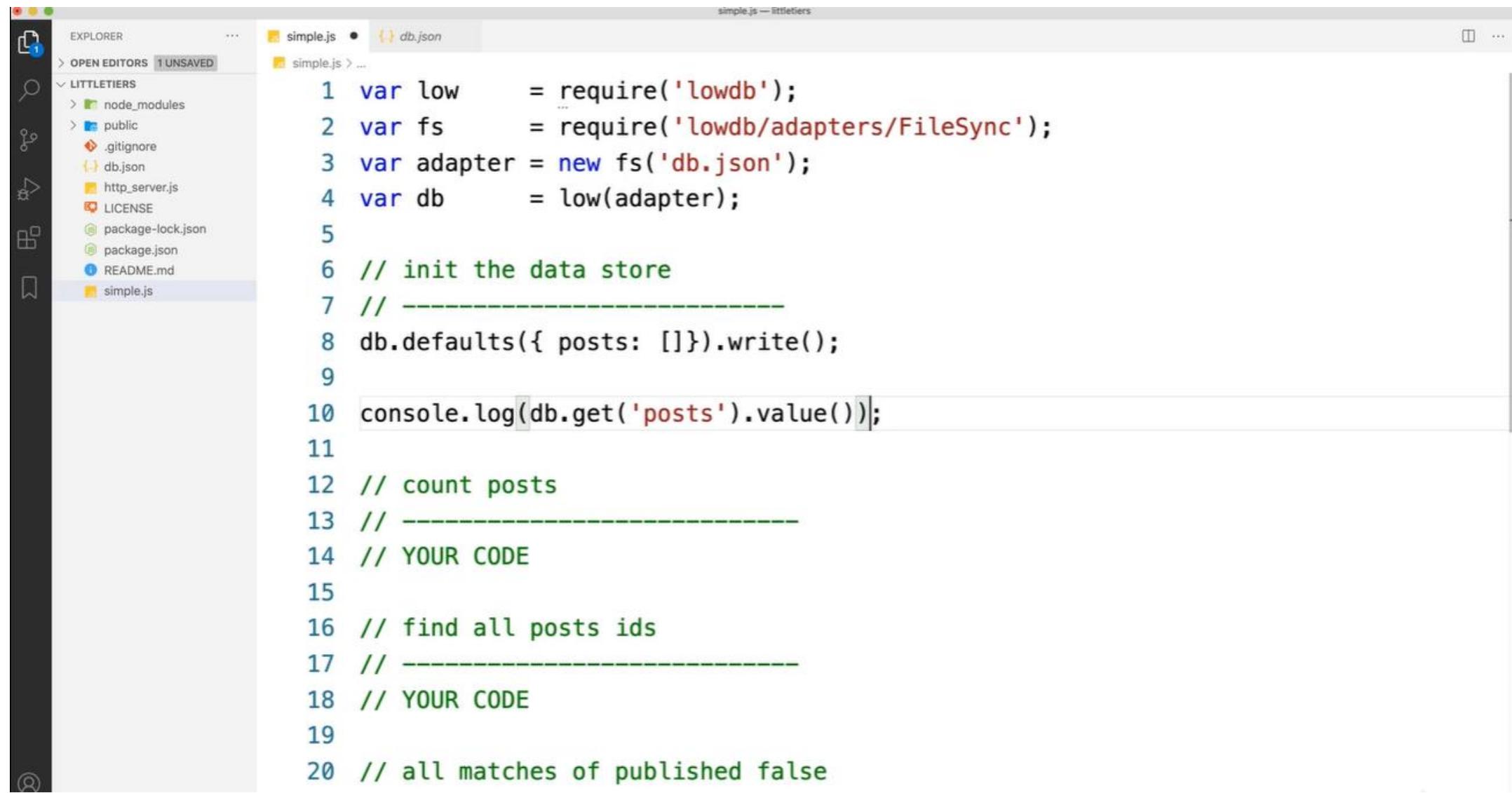


The screenshot shows a code editor interface with a dark theme. On the left is the Explorer sidebar, which lists files and folders for a project named 'LITTLETIERS'. The 'simple.js' file is selected. The main editor area contains the following code:

```
simple.js — littletiers
simple.js > ...
7 // -----
8 db.defaults({ posts: [] }).write();
9
10 // Add a post
11
12 db.get('posts')
13 .push({ id: 2, title: 'great', published: true })
14 .write()
15 db.get('posts')
16 .push({ id: 3, title: 'new own', published: false })
17 .write()
18 db.get('posts')
19 .push({ id: 4, title: 'random', published: false })
20 .write()
21
22
23 // count posts
24 // -----
25 // YOUR CODE
26
```

The code uses the `db` object to interact with a database. It starts by initializing the database with an empty array of posts. Then, it adds three posts with IDs 2, 3, and 4, setting their titles and publishing status. Finally, it adds a comment indicating where the user's code should be placed.

# Creating Little DB (21/22)



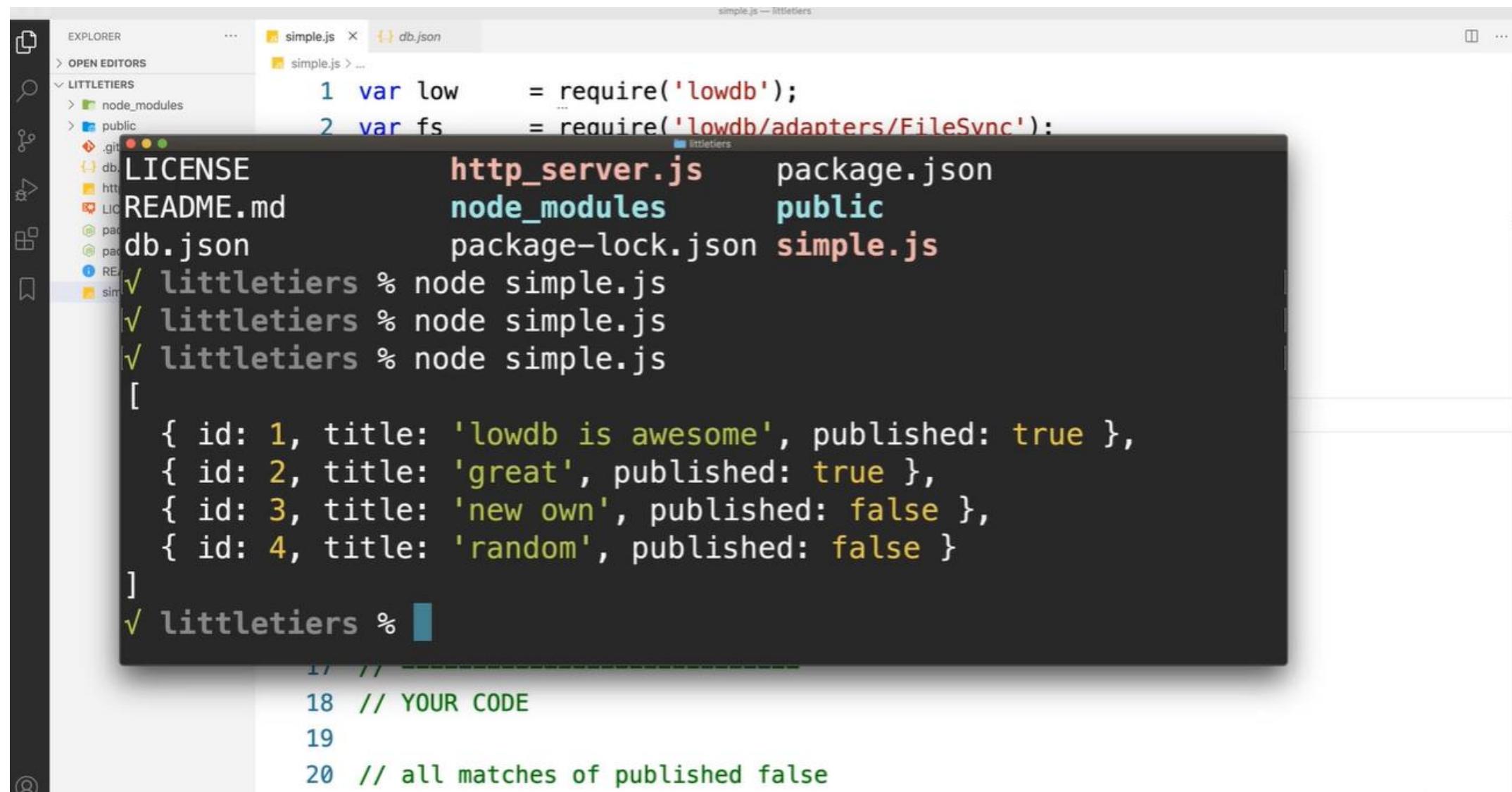
The screenshot shows a code editor interface with the following details:

- Explorer Bar:** On the left, it shows a tree view of the project structure under "LITTLETIERS". The files listed are: node\_modules, public, .gitignore, db.json, http\_server.js, LICENSE, package-lock.json, package.json, README.md, and simple.js.
- Editor Area:** The main area displays the content of the "simple.js" file. The code uses the `low` library to interact with a database stored in `db.json`. The code includes comments for initializing the data store, counting posts, and finding all post IDs.
- File Status:** At the top right, it says "simple.js — littletiers" and "1 UNSAVED".

```
simple.js — littletiers
simple.js • db.json
simple.js > ...

1 var low      = require('lowdb');
2 var fs       = require('lowdb/adapters/FileSync');
3 var adapter = new fs('db.json');
4 var db       = low(adapter);
5
6 // init the data store
7 // -----
8 db.defaults({ posts: [] }).write();
9
10 console.log(db.get('posts').value());
11
12 // count posts
13 // -----
14 // YOUR CODE
15
16 // find all posts ids
17 // -----
18 // YOUR CODE
19
20 // all matches of published false
```

# Creating Little DB (22/22)



The screenshot shows a Visual Studio Code interface with the following details:

- Explorer View:** Shows a file tree for a project named "littletiers". The tree includes files like LICENSE, README.md, db.json, http\_server.js, package.json, node\_modules, public, package-lock.json, simple.js, and a .git folder.
- Terminal View:** Displays the command-line output of running the "simple.js" script multiple times. The output shows the creation of a database with four entries:

id	title	published
1	'lowdb is awesome'	true
2	'great'	true
3	'new own'	false
4	'random'	false

- Code Editor View:** Shows the source code for "simple.js". The code uses the "lowdb" library to interact with a database. It defines variables "low" and "fs" and then loops through the database to print each entry's title and published status.

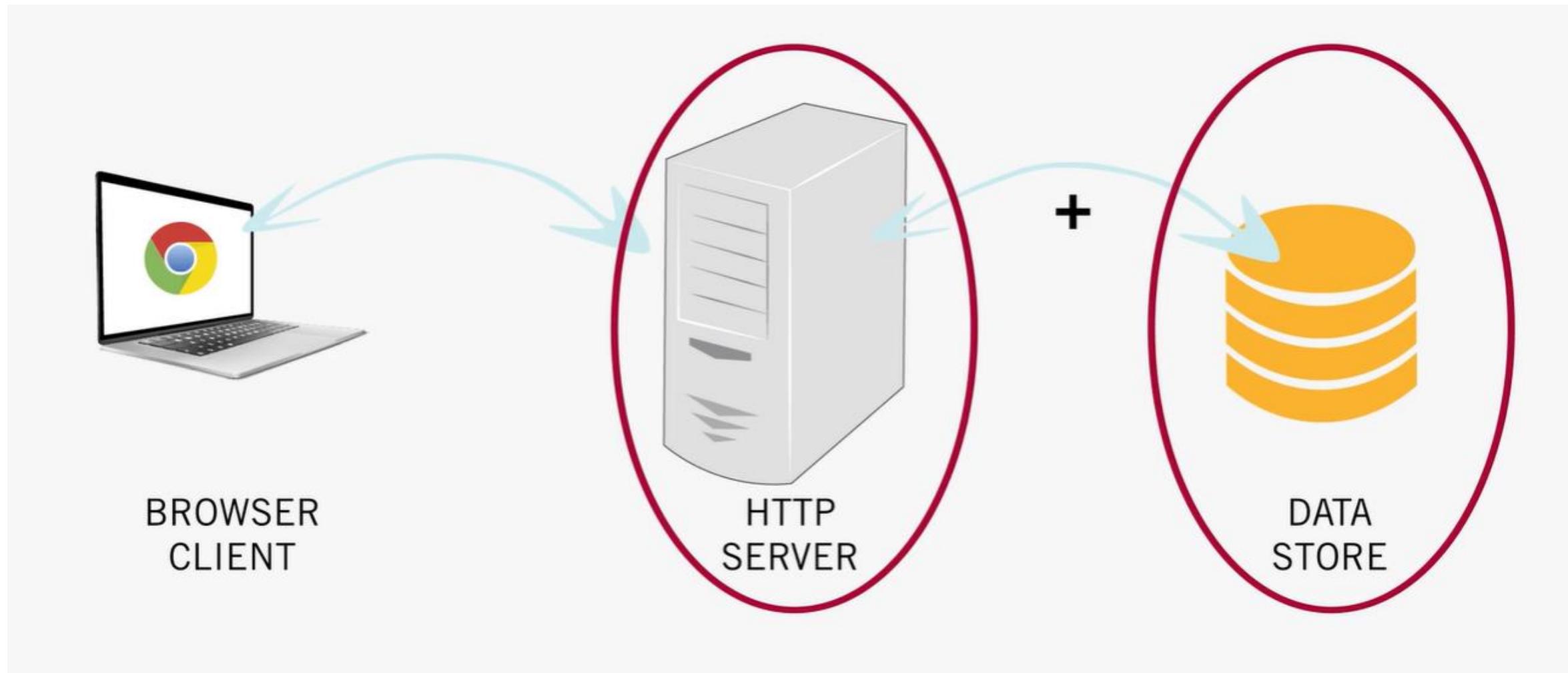
```
simple.js — littletiers
1 var low      = require('lowdb');
2 var fs       = require('lowdb/adapters/FileSync');

LICENCE          http_server.js    package.json
README.md        node_modules      public
db.json          package-lock.json simple.js

littletiers % node simple.js
littletiers % node simple.js
littletiers % node simple.js
[{"id": 1, "title": "lowdb is awesome", "published": true}, {"id": 2, "title": "great", "published": true}, {"id": 3, "title": "new own", "published": false}, {"id": 4, "title": "random", "published": false}]
littletiers %

17 // -----
18 // YOUR CODE
19
20 // all matches of published false
```

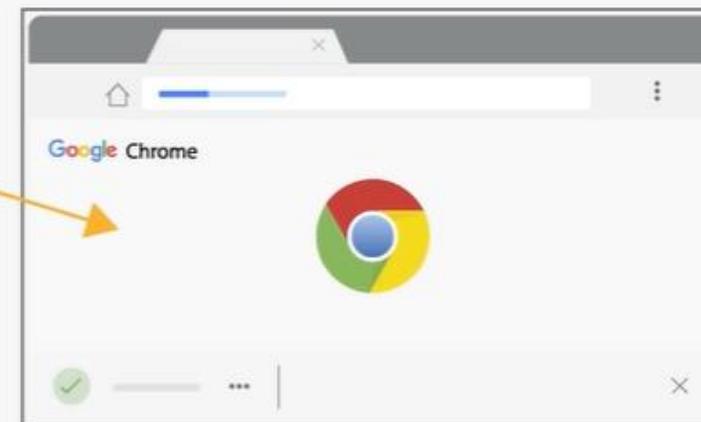
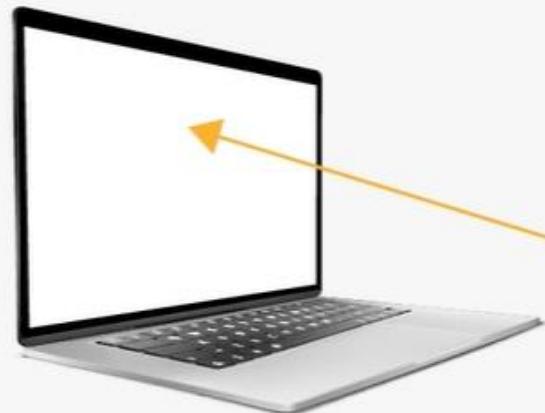
# Three Tiers: Client, Server, & Storage



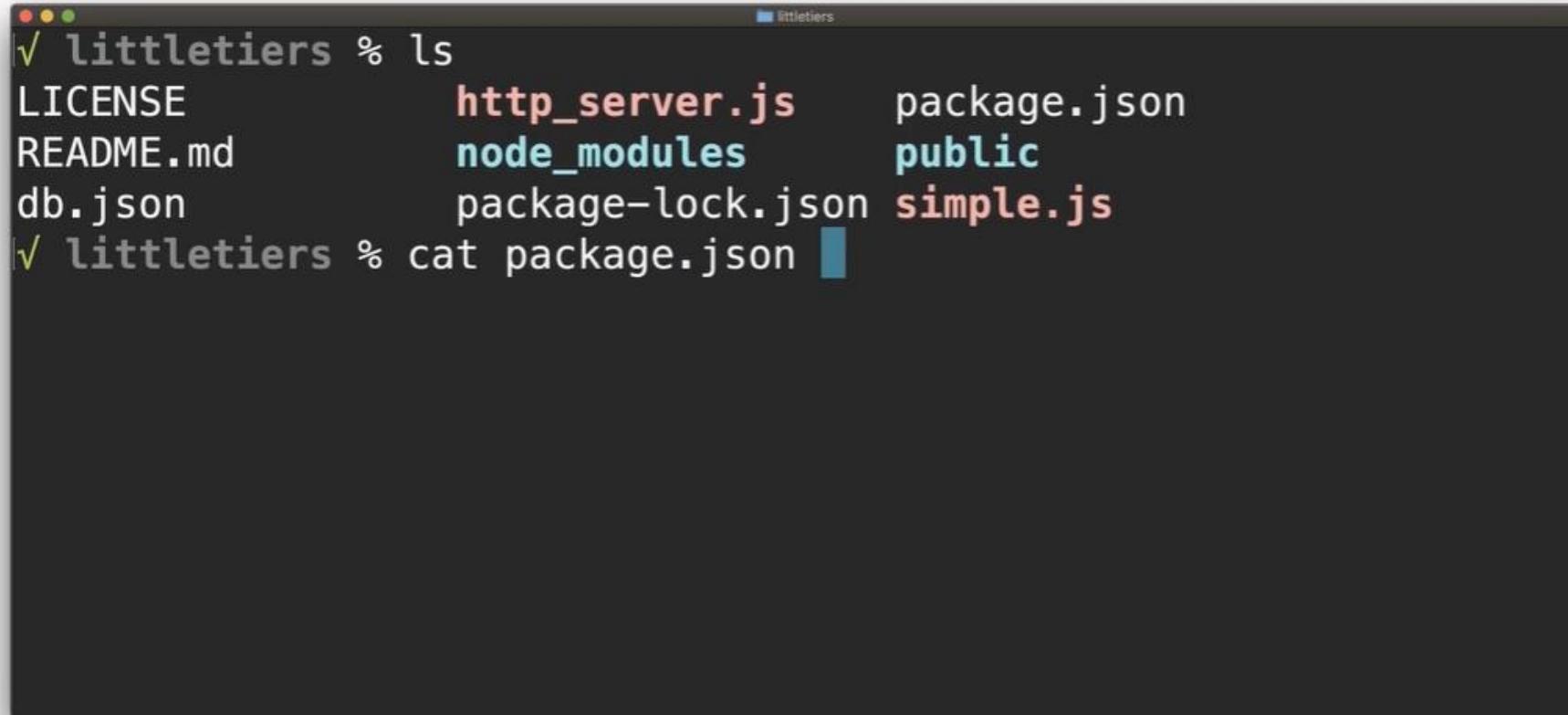
# Web Server + Data Store

LowDB

Express



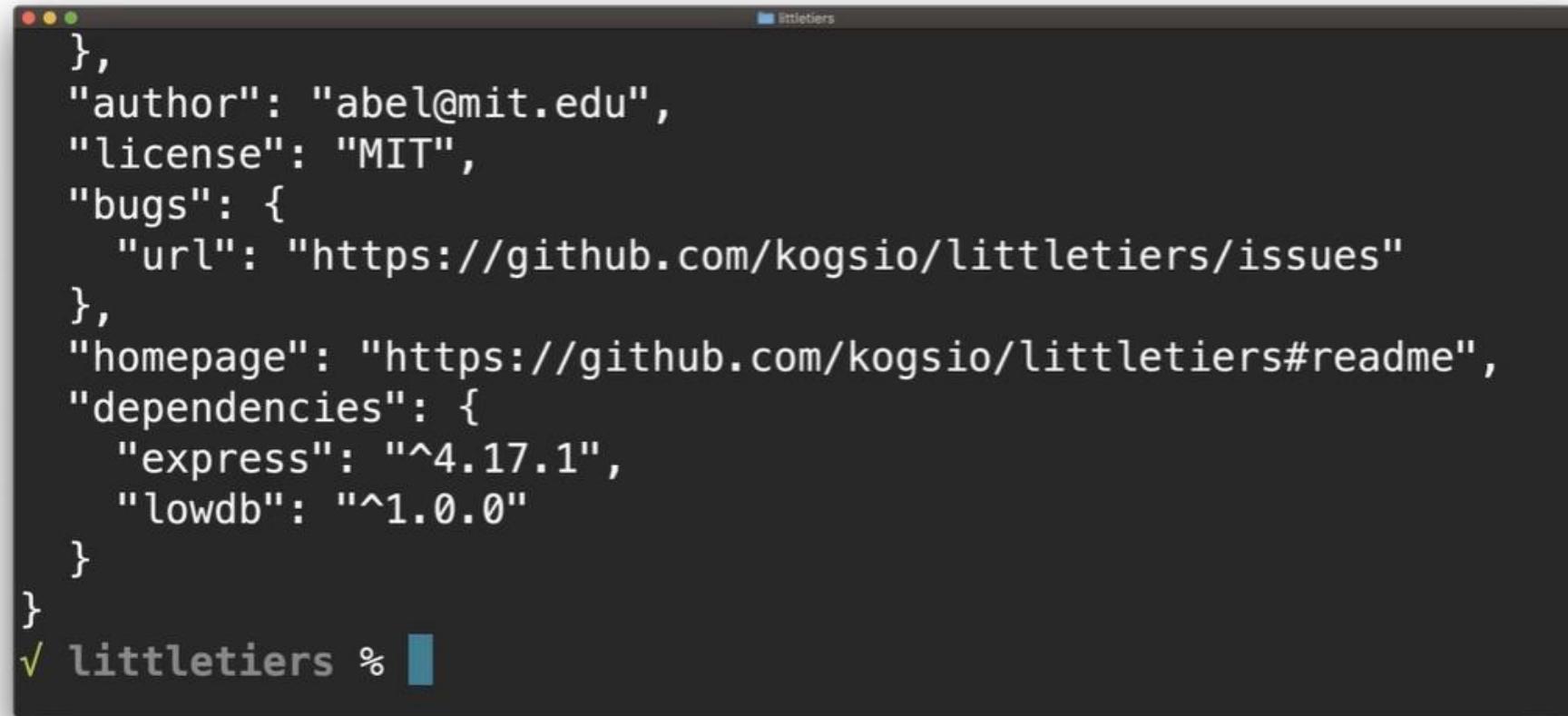
# Creating Server Plus Data Storage (1/23)



A screenshot of a macOS terminal window titled "littletiers". The window shows the output of the "ls" command, listing files: LICENSE, README.md, db.json, http\_server.js, node\_modules, package-lock.json, package.json, public, and simple.js. Below this, the command "cat package.json" is entered.

```
littletiers % ls
LICENSE          http_server.js    package.json
README.md        node_modules      public
db.json          package-lock.json simple.js
littletiers % cat package.json
```

## Creating Server Plus Data Storage (2/23)



A screenshot of a terminal window titled "littletiers". The window displays a JSON object representing a package's metadata. The content includes fields for author ("author": "abel@mit.edu"), license ("license": "MIT"), bugs ("bugs": { "url": "https://github.com/kogsio/littletiers/issues" }), homepage ("homepage": "https://github.com/kogsio/littletiers#readme"), and dependencies ("dependencies": { "express": "^4.17.1", "lowdb": "^1.0.0" }). The terminal prompt at the bottom is "littletiers %".

```
},  
  "author": "abel@mit.edu",  
  "license": "MIT",  
  "bugs": {  
    "url": "https://github.com/kogsio/littletiers/issues"  
  },  
  "homepage": "https://github.com/kogsio/littletiers#readme",  
  "dependencies": {  
    "express": "^4.17.1",  
    "lowdb": "^1.0.0"  
  }  
}  
littletiers %
```

## Creating Server Plus Data Storage (3/23)

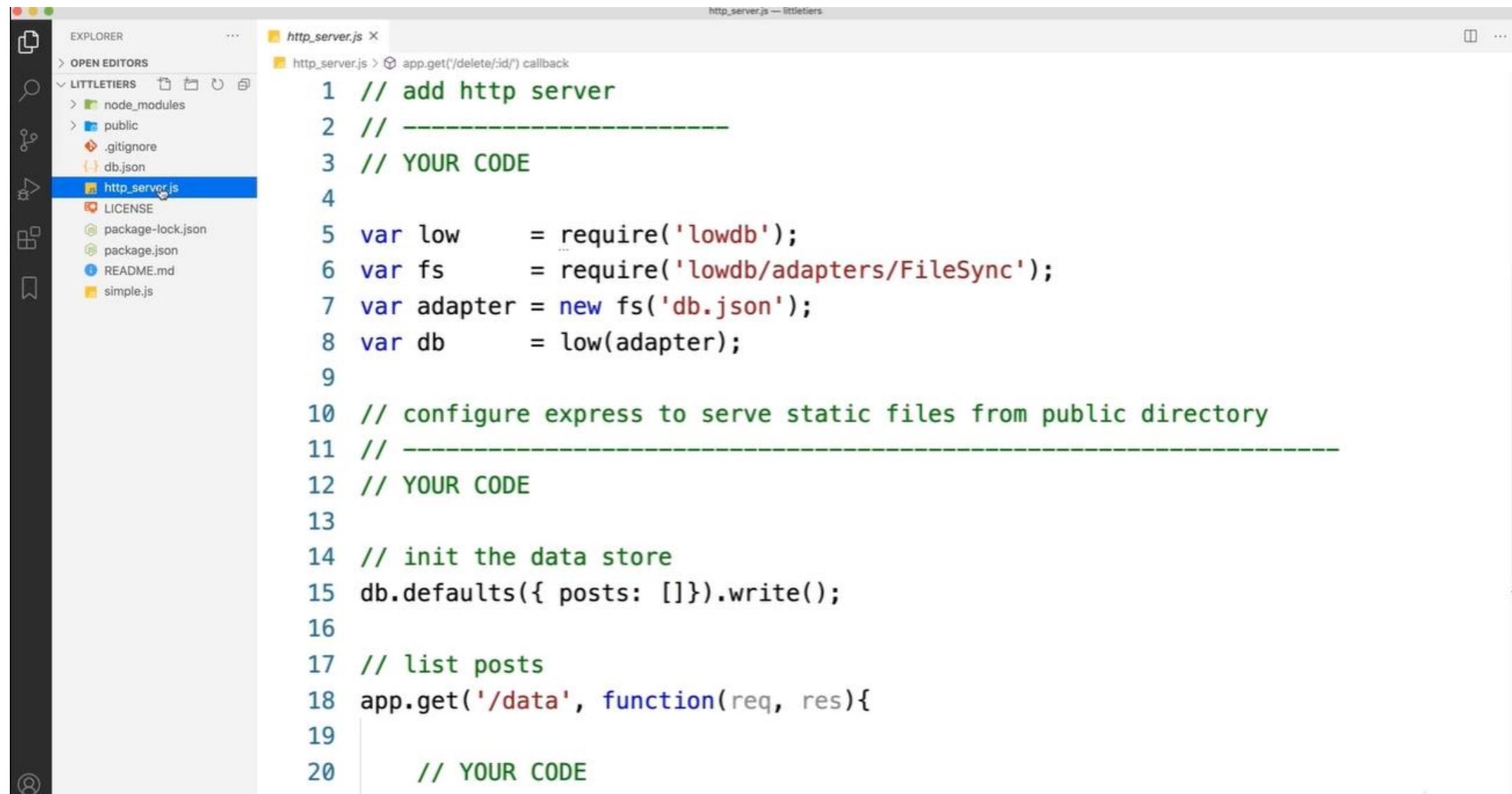
```
audited 56 packages in 0.79s
found 0 vulnerabilities

✓ littletiers % npm install lowdb
+ lowdb@1.0.0
updated 1 package and audited 56 packages in 0.858s
found 0 vulnerabilities

✓ littletiers % npm install
audited 56 packages in 0.71s
found 0 vulnerabilities

✓ littletiers %
```

# Creating Server Plus Data Storage (4/23)



```
http_server.js — littletiers
http_server.js ×
http_server.js > app.get('/delete/:id') callback

1 // add http server
2 // -----
3 // YOUR CODE
4
5 var low      = require('lowdb');
6 var fs       = require('lowdb/adapters/FileSync');
7 var adapter = new fs('db.json');
8 var db       = low(adapter);
9
10 // configure express to serve static files from public directory
11 // -----
12 // YOUR CODE
13
14 // init the data store
15 db.defaults({ posts: [] }).write();
16
17 // list posts
18 app.get('/data', function(req, res){
19   // YOUR CODE
20 })
```

# Creating Server Plus Data Storage (5/23)



The screenshot shows a Microsoft Visual Studio Code (VS Code) interface. On the left is the Explorer sidebar, which lists the project structure under 'Littletiers'. The 'http\_server.js' file is open in the main editor area. The code is a Node.js application for a REST API. It includes logic for handling posts and published status. Several curl command examples are provided in the code to test the API endpoints.

```
26 //     curl http://localhost:3000/posts/ping/1/false
27 // -----
28 app.get('/posts/:title/:id/:published', function(req, res){
29
30     // YOUR CODE
31
32 });
33
34 // -----
35 // filter by published state - test using:
36 //     curl http://localhost:3000/published/true
37 // -----
38 app.get('/published/:boolean', function(req, res){
39
40     // YOUR CODE
41
42 });
43
44 // -----
45 // update published value - test using:
```

# Creating Server Plus Data Storage (6/23)

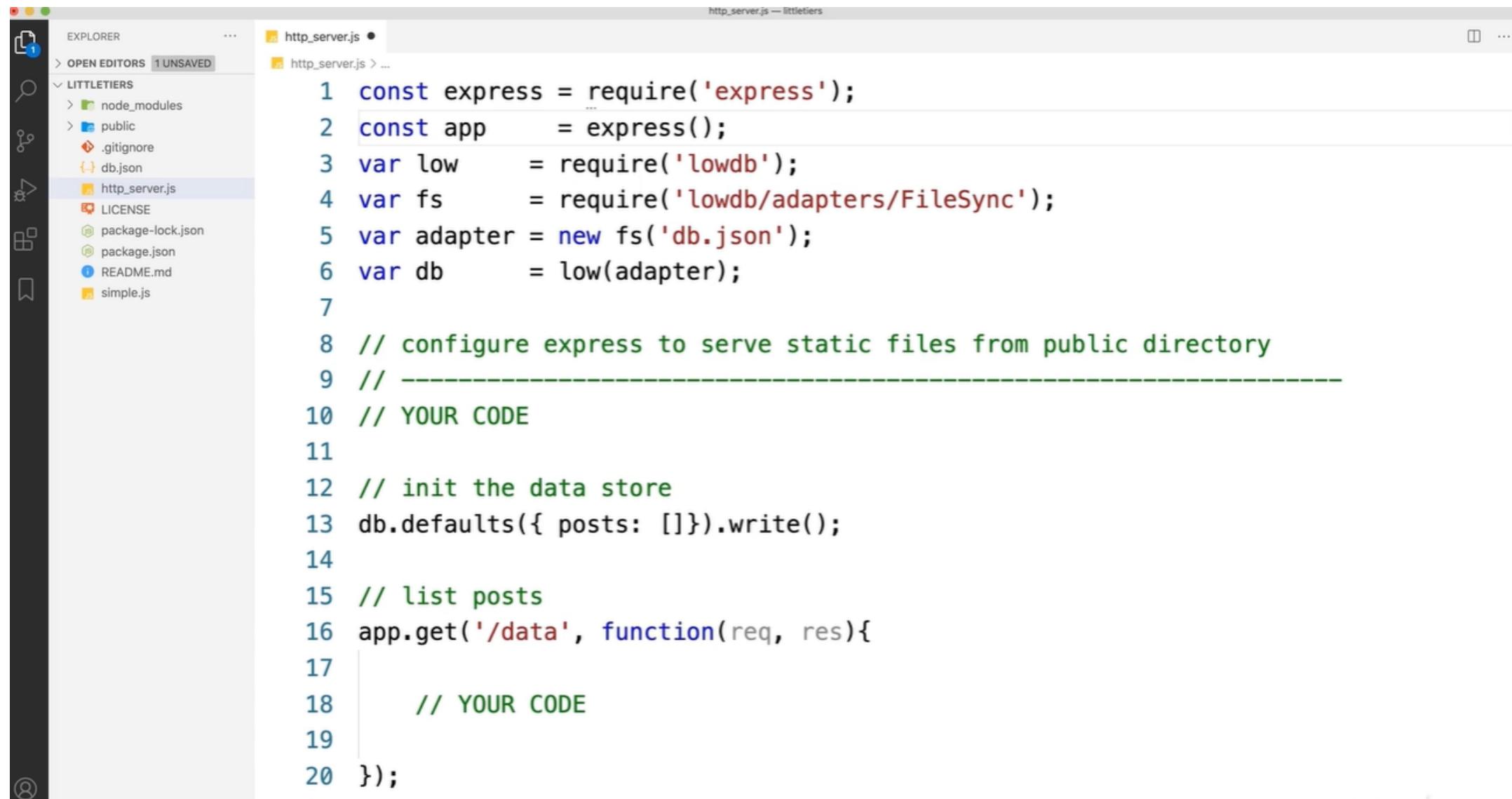


The screenshot shows a code editor interface with the following details:

- Explorer Bar:** On the left, it shows a tree view of the project structure under "LITTLETIERS". The files listed are: node\_modules, public, .gitignore, db.json, http\_server.js (which is selected), LICENSE, package-lock.json, package.json, README.md, and simple.js.
- Code Editor:** The main area displays the `http_server.js` file with the following content:

```
58 app.get('/delete/:id/', function(req, res){  
59     // YOUR CODE  
60 };  
61  
62 // start server  
63 // -----  
64 // YOUR CODE  
65  
66 //-----  
67
```

# Creating Server Plus Data Storage (7/23)

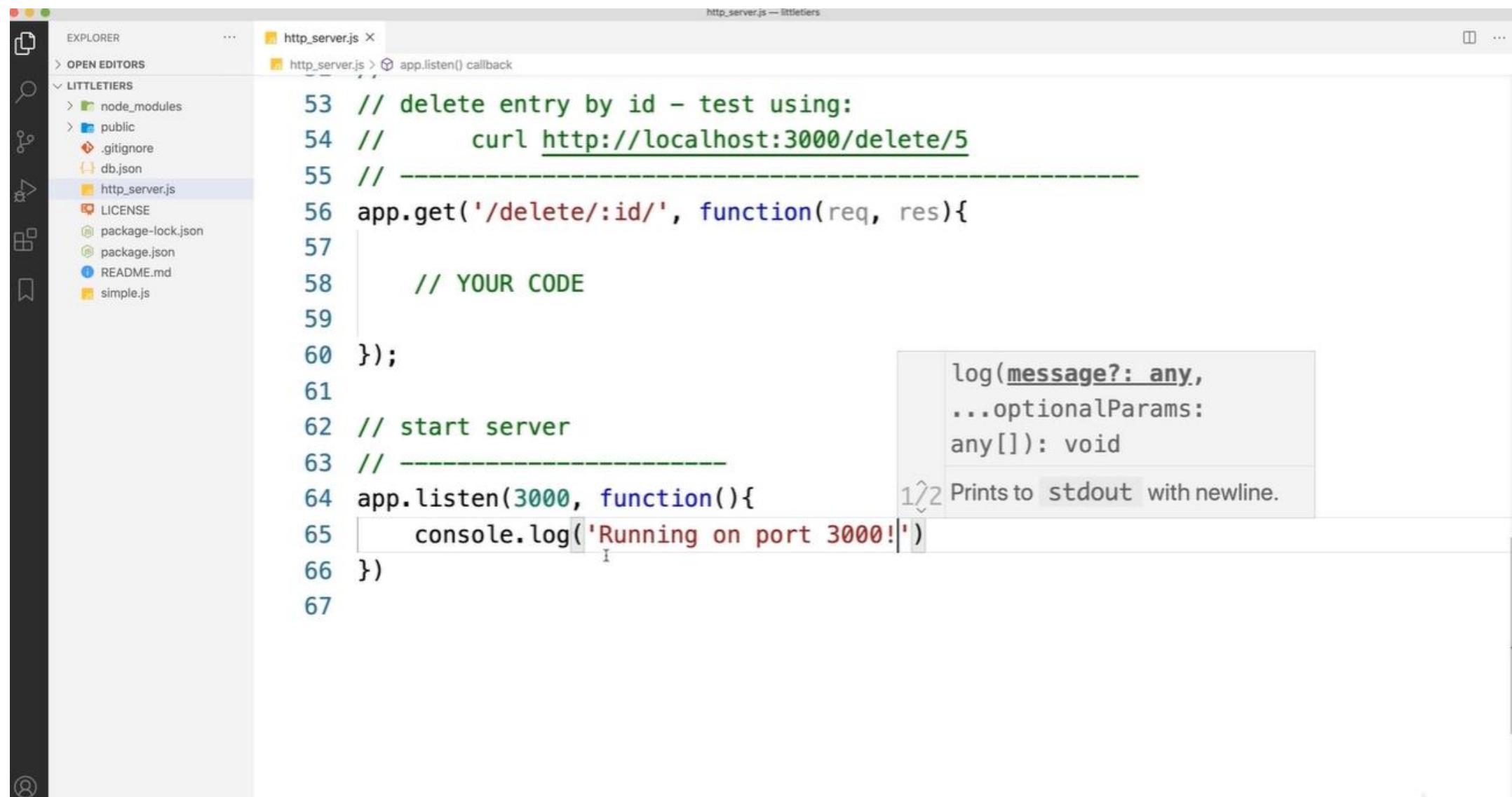


The screenshot shows a code editor interface with the following details:

- Explorer Bar:** On the left, it shows a tree view of the project structure under "LITTLETIERS". The files listed are: node\_modules, public, .gitignore, db.json, http\_server.js (which is selected), LICENSE, package-lock.json, package.json, README.md, and simple.js.
- Code Editor:** The main area displays the content of the http\_server.js file. The code is written in Node.js and uses Express and lowdb libraries.
- Code Content:**

```
1 const express = require('express');
2 const app = express();
3 var low = require('lowdb');
4 var fs = require('lowdb/adapters/FileSync');
5 var adapter = new fs('db.json');
6 var db = low(adapter);
7
8 // configure express to serve static files from public directory
9 // -----
10 // YOUR CODE
11
12 // init the data store
13 db.defaults({ posts: [] }).write();
14
15 // list posts
16 app.get('/data', function(req, res){
17
18     // YOUR CODE
19
20});
```

# Creating Server Plus Data Storage (8/23)



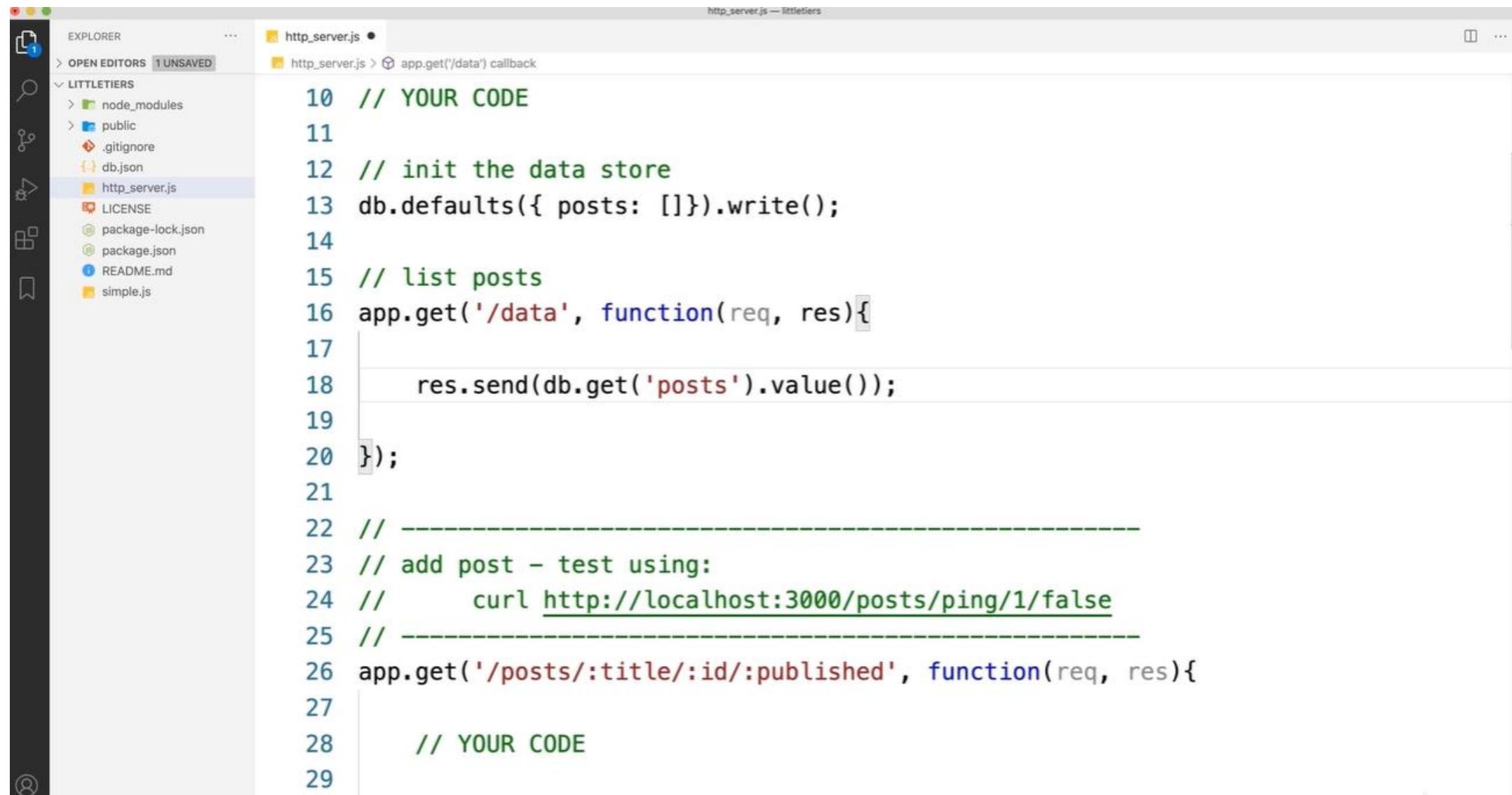
The screenshot shows a Microsoft Visual Studio Code (VS Code) interface. On the left is the Explorer sidebar with a tree view of a project named 'LITTLETIERS'. The tree includes 'node\_modules', 'public', '.gitignore', 'db.json', 'http\_server.js' (which is currently selected), 'LICENSE', 'package-lock.json', 'package.json', 'README.md', and 'simple.js'. The main editor area displays the 'http\_server.js' code:

```
53 // delete entry by id - test using:  
54 // curl http://localhost:3000/delete/5  
55 // -----  
56 app.get('/delete/:id/', function(req, res){  
57  
58     // YOUR CODE  
59  
60});  
61  
62 // start server  
63 // -----  
64 app.listen(3000, function(){  
65     console.log('Running on port 3000!')  
66 })  
67
```

A tooltip is visible over the 'console.log' call at line 65, providing documentation for the Node.js `log` function:

log(message?: any,  
...optionalParams:  
any[]): void  
172 Prints to stdout with newline.

# Creating Server Plus Data Storage (9/23)



The screenshot shows a code editor interface with the following details:

- Explorer Bar:** On the left, it shows a file tree for a project named "LITTLETIERS". The files listed are: node\_modules, public, .gitignore, db.json, http\_server.js (which is currently selected), LICENSE, package-lock.json, package.json, README.md, and simple.js.
- Editor Area:** The main area displays a Node.js script named "http\_server.js". The code is color-coded and numbered from 10 to 29.

```
10 // YOUR CODE
11
12 // init the data store
13 db.defaults({ posts: []}).write();
14
15 // list posts
16 app.get('/data', function(req, res){
17
18     res.send(db.get('posts').value());
19
20 });
21
22 // -----
23 // add post - test using:
24 //     curl http://localhost:3000/posts/ping/1/false
25 // -----
26 app.get('/posts/:title/:id/:published', function(req, res){
27
28     // YOUR CODE
29 }
```

# Creating Server Plus Data Storage (10/23)

The screenshot shows a Microsoft Visual Studio Code interface. On the left is the Explorer sidebar with a tree view of a project named 'LITTLETIERS'. The tree includes 'node\_modules', 'public', '.git', 'db', 'http', 'LICENSE', 'README.md', 'db.json', 'package.json', 'package-lock.json', 'simple.js', and 'RENAME'. The main area is a terminal window with the following output:

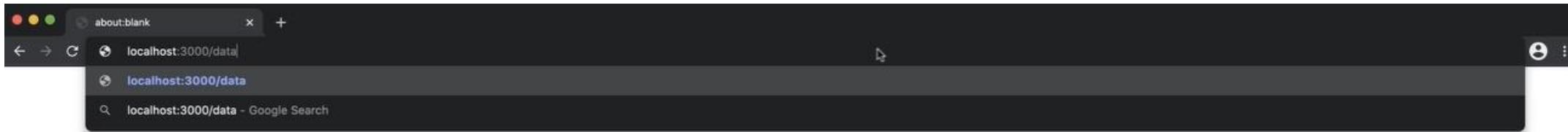
```
10 // YOUR CODE
11
found 0 vulnerabilities
✓ littletiers % npm install
audited 56 packages in 0.71s
found 0 vulnerabilities

✓ littletiers % ls
LICENSE          http_server.js      package.json
README.md        node_modules       public
db.json          package-lock.json simple.js
✓ littletiers % node http_server.js
Running on port 3000!
```

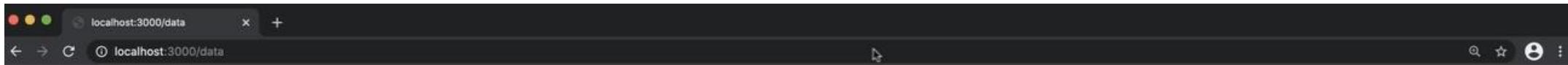
Below the terminal, the code editor shows the file 'http\_server.js' with the following content:

```
26 app.get('/posts/:title/:id/:published', function(req, res){
27
28     // YOUR CODE
29 }
```

# Creating Server Plus Data Storage (11/23)



# Creating Server Plus Data Storage (12/23)



```
[ {"id":1,"title":"lowdb is awesome","published":true}, {"id":2,"title":"great","published":true}, {"id":3,"title":"new own","published":false}, {"id":4,"title":"random","published":false} ]
```

# Creating Server Plus Data Storage (13/23)

```
http_server.js -- littletiers
22 // -----
23 // add post - test using:
24 //     curl http://localhost:3000/posts/ping/1/false
25 // -----
26 app.get('/posts/:title/:id/:published', function(req, res){
27
28     // YOUR CODE
29
30 });
31
32 // -----
33 // filter by published state - test using:
34 //     curl http://localhost:3000/published/true
35 // -----
36 app.get('/published/:boolean', function(req, res){
37
38     // YOUR CODE
39
40 });
41
```

# URL Routes

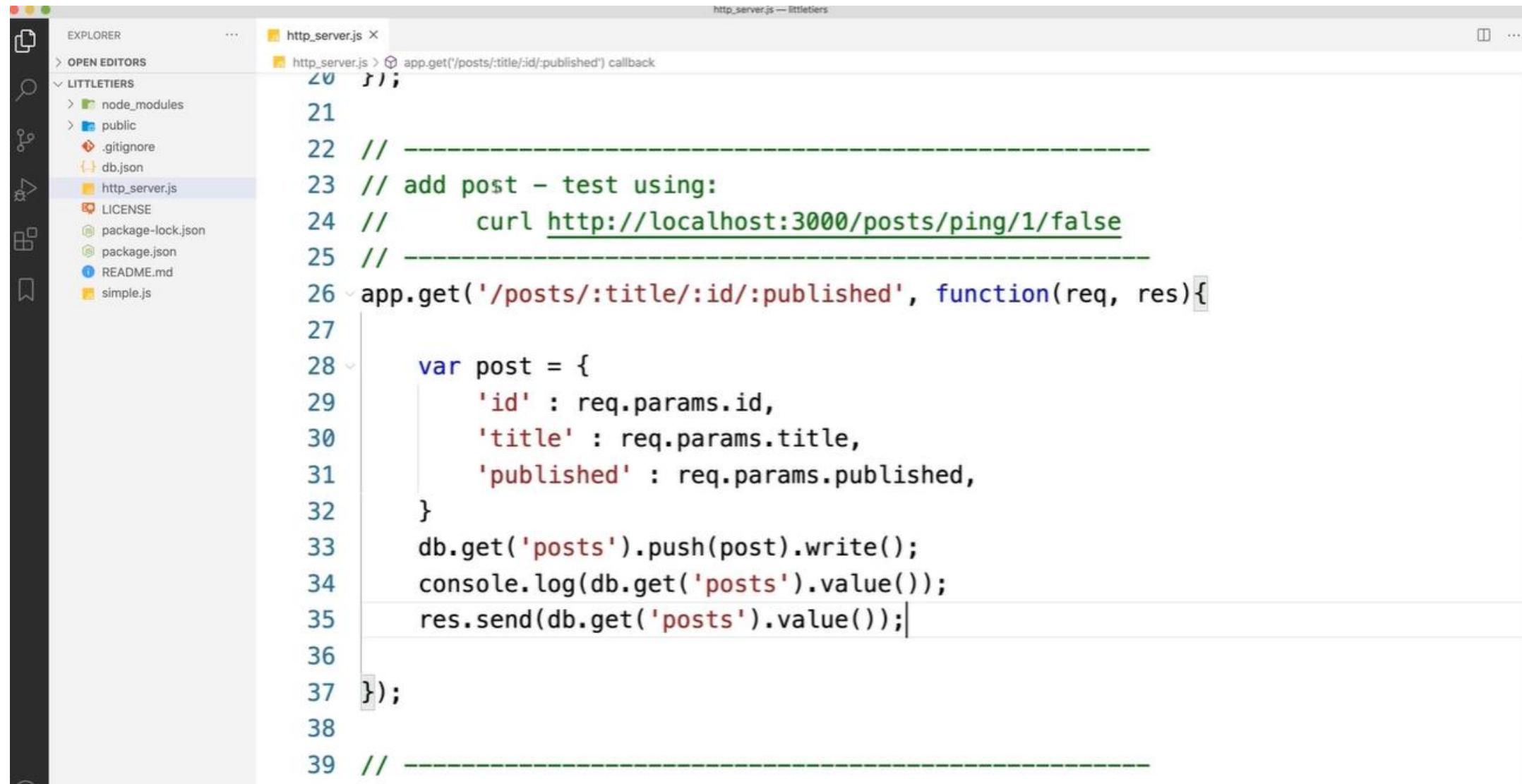
route      title      id      id

```
app.get('/posts/:title/:id/:published', function(req, res) {  
    // function implementations  
});
```

URL      route      title      id      published

**http://localhost:3000/post>Hello/9/false**

# Creating Server Plus Data Storage (15/23)

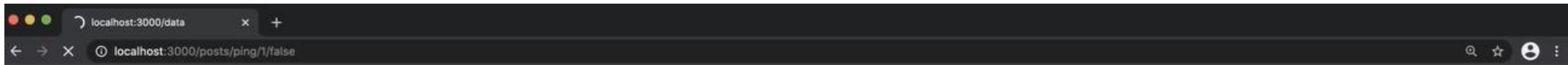


The screenshot shows a Microsoft Visual Studio Code (VS Code) interface. On the left is the Explorer sidebar, which lists the project structure under 'LITTLETIERS'. The 'http\_server.js' file is selected in the list. The main editor area displays the code for this file. The code is a Node.js script that defines an HTTP endpoint for adding posts. It uses the 'app.get' method to handle requests to '/posts/:title/:id/:published'. The response is generated by creating a new 'post' object from the request parameters and pushing it to a database ('db'). The code includes comments and a curl command to test the endpoint.

```
http_server.js — littletiers
http_server.js X
http_server.js > app.get('/posts/:title/:id/:published') callback

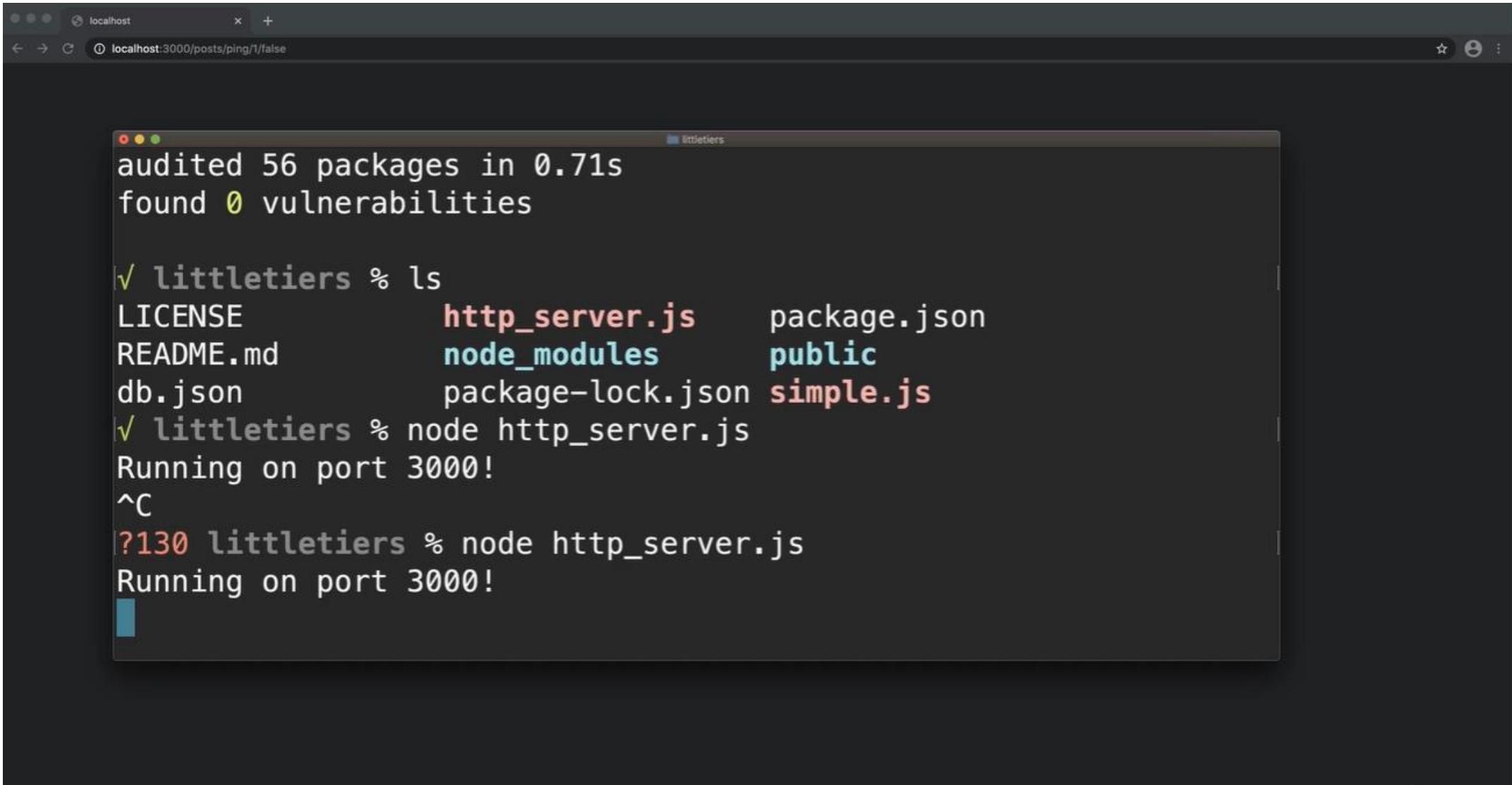
20 // -----
21
22 // -----
23 // add post - test using:
24 //     curl http://localhost:3000/posts/ping/1/false
25 // -----
26 app.get('/posts/:title/:id/:published', function(req, res){
27
28     var post = {
29         'id' : req.params.id,
30         'title' : req.params.title,
31         'published' : req.params.published,
32     }
33     db.get('posts').push(post).write();
34     console.log(db.get('posts').value());
35     res.send(db.get('posts').value());
36
37 });
38
39 // -----
```

## Creating Server Plus Data Storage (16/23)



```
[ {"id":1,"title":"lowdb is awesome","published":true}, {"id":2,"title":"great","published":true}, {"id":3,"title":"new own","published":false}, {"id":4,"title":"random","published":false} ]
```

# Creating Server Plus Data Storage (17/23)



A screenshot of a terminal window titled "littletiers". The terminal shows the following sequence of commands and output:

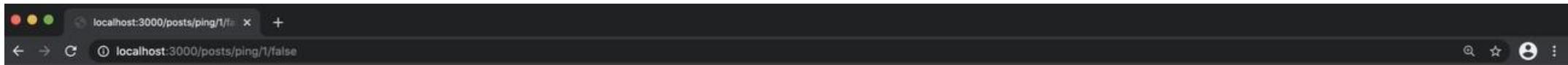
```
audited 56 packages in 0.71s
found 0 vulnerabilities

✓ littletiers % ls
LICENSE          http_server.js    package.json
README.md        node_modules      public
db.json          package-lock.json simple.js

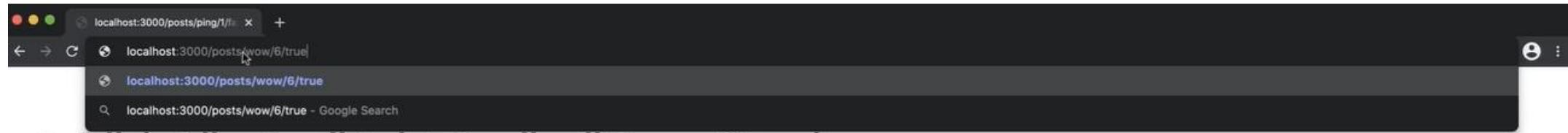
✓ littletiers % node http_server.js
Running on port 3000!
^C
?130 littletiers % node http_server.js
Running on port 3000!
```

The terminal window is set against a dark background and has a blue header bar. Above the terminal, a browser window is visible, showing the URL "localhost:3000/posts/ping/1/false".

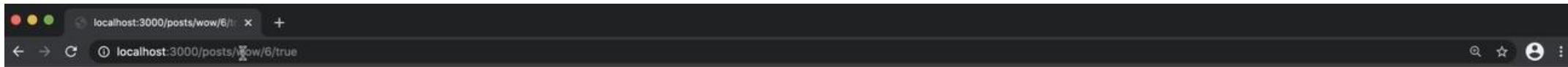
## Creating Server Plus Data Storage (18/23)



# Creating Server Plus Data Storage (19/23)

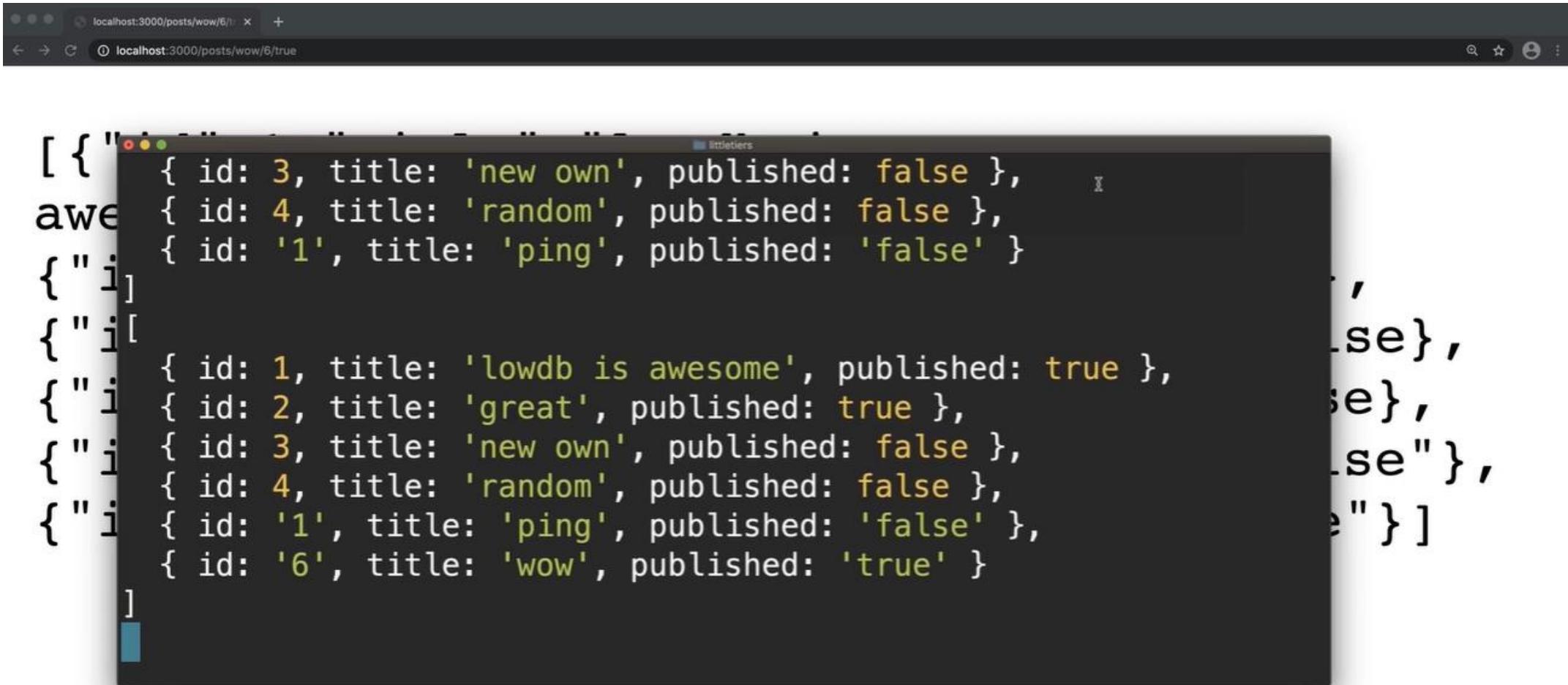


## Creating Server Plus Data Storage (20/23)



```
[ {"id":1,"title":"lowdb is awesome","published":true}, {"id":2,"title":"great","published":true}, {"id":3,"title":"new own","published":false}, {"id":4,"title":"random","published":false}, {"id":1,"title":"ping","published":false}, {"id":6,"title":"wow","published":true} ]
```

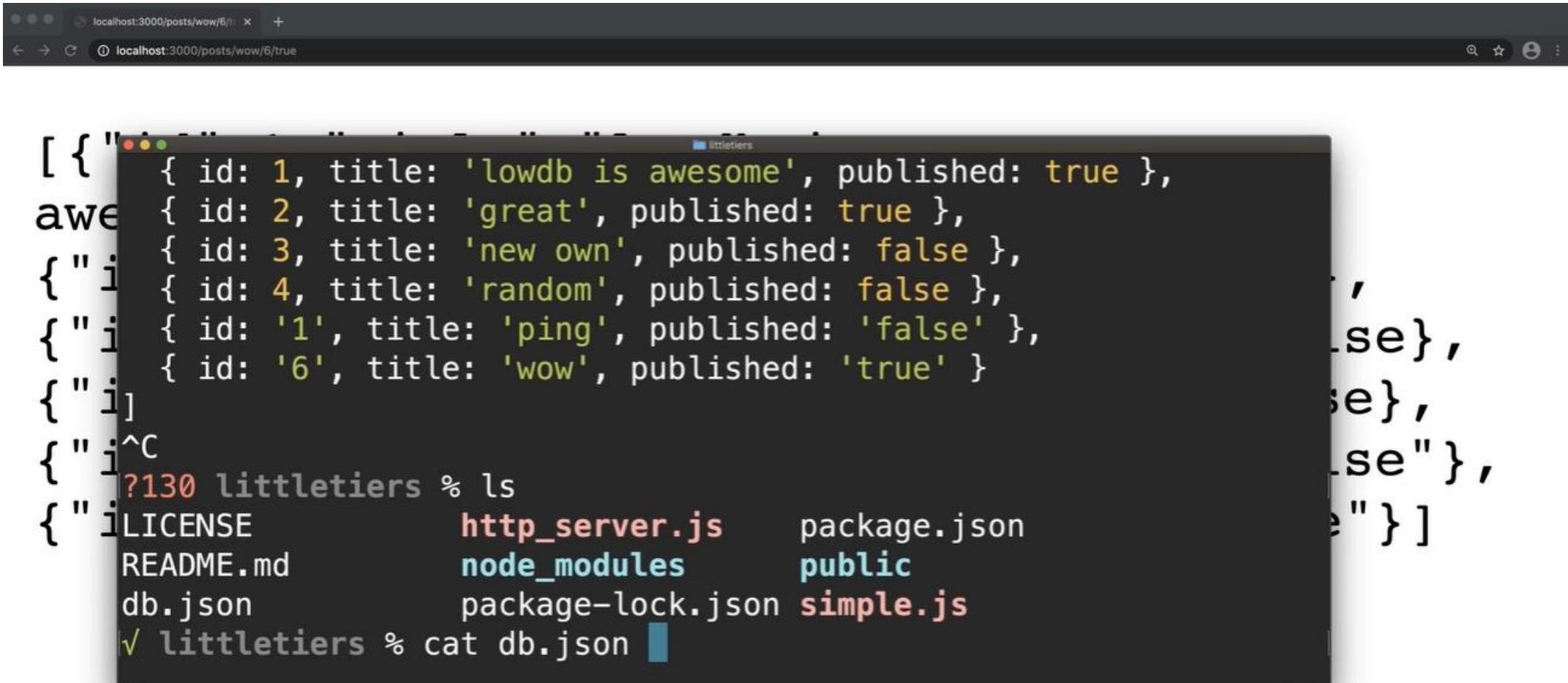
# Creating Server Plus Data Storage (21/23)



A screenshot of a web browser window displaying a JSON response. The URL in the address bar is `localhost:3000/posts-wow/6/true`. The page content shows a large array of objects representing posts. The objects have properties: id, title, and published. Some posts are published (true) and some are not (false). The titles include "new own", "random", "ping", and "wow". The JSON structure is deeply nested, with multiple levels of arrays and objects.

```
[ { "id": 3, "title": "new own", "published": false }, { "id": 4, "title": "random", "published": false }, { "id": "1", "title": "ping", "published": "false" } ] [ { "id": 1, "title": "lowdb is awesome", "published": true }, { "id": 2, "title": "great", "published": true }, { "id": 3, "title": "new own", "published": false }, { "id": 4, "title": "random", "published": false }, { "id": "1", "title": "ping", "published": "false" }, { "id": "6", "title": "wow", "published": "true" } ]
```

# Creating Server Plus Data Storage (22/23)



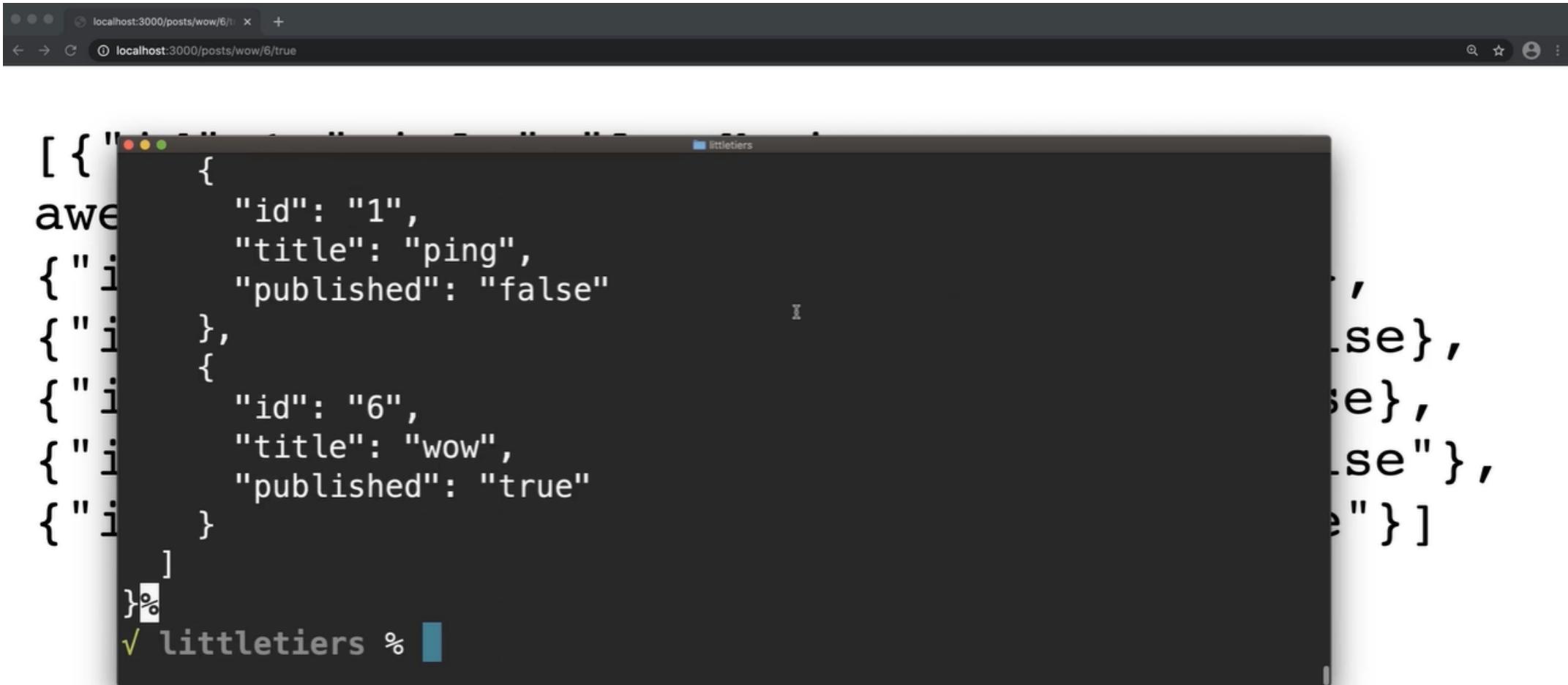
A screenshot of a terminal window titled "littletiers". The window shows a JSON object representing a collection of posts:

```
[ { "id": 1, "title": "lowdb is awesome", "published": true },  
  { "id": 2, "title": "great", "published": true },  
  { "id": 3, "title": "new own", "published": false },  
  { "id": 4, "title": "random", "published": false },  
  { "id": "1", "title": "ping", "published": false },  
  { "id": "6", "title": "wow", "published": true } ]
```

The terminal also displays the output of the `ls` command, showing files like `LICENSE`, `http_server.js`, `node_modules`, `package.json`, `public`, `db.json`, `package-lock.json`, and `simple.js`.

```
?130 littletiers % ls  
LICENSE          http_server.js    package.json  
README.md        node_modules      public  
db.json          package-lock.json simple.js  
littletiers % cat db.json
```

# Creating Server Plus Data Storage (23/23)



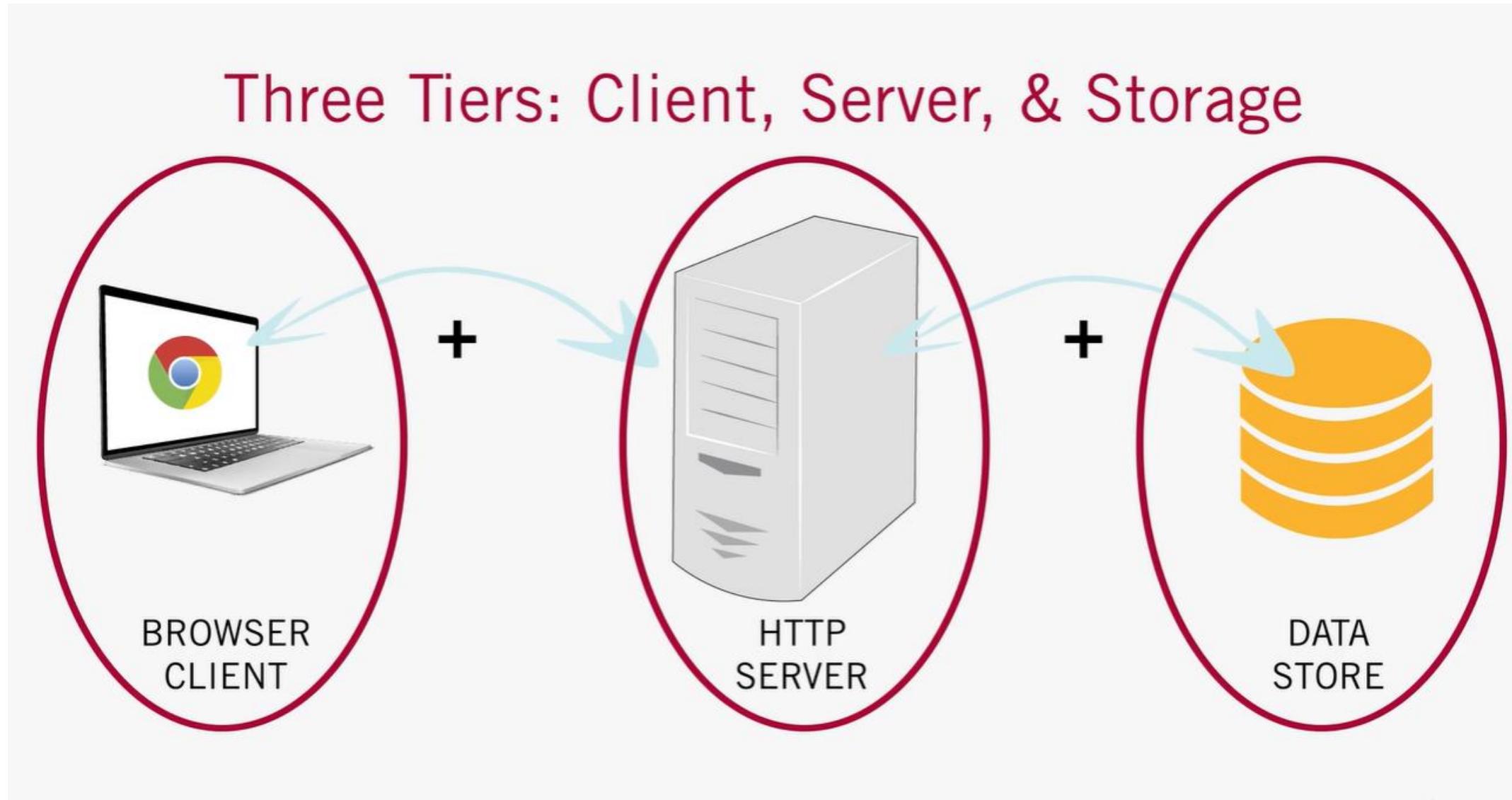
A screenshot of a terminal window titled "littletiers". The window shows a command being typed into the terminal:

```
[ { "id": "1",  
  "title": "ping",  
  "published": "false"  
,  
{ "id": "2",  
  "title": "pong",  
  "published": "false"  
,  
{ "id": "3",  
  "title": "pong",  
  "published": "false"  
,  
{ "id": "4",  
  "title": "pong",  
  "published": "false"  
,  
{ "id": "5",  
  "title": "pong",  
  "published": "false"  
,  
  } ]
```

The terminal prompt is "littletiers %". Above the terminal window, a browser window is visible with the URL "localhost:3000/posts-wow/6/true".

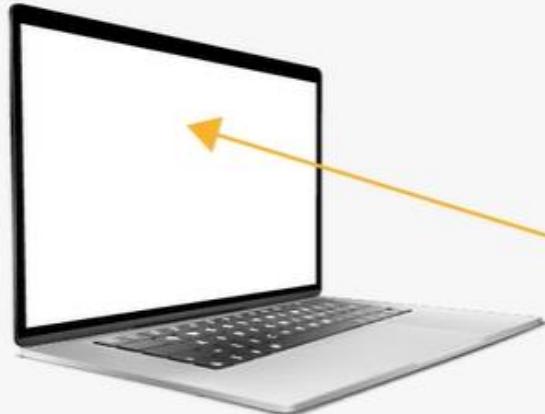
<https://expressjs.com/en/guide/routing.html>

# Three Tiers: Client, Server, & Storage



# Web Server + Data Store + UI

LowDB  
Express  
  

Bootstrap

# Superagent Helps You Make HTTP Calls



Superagent

<https://www.npmjs.com/package/superagent>



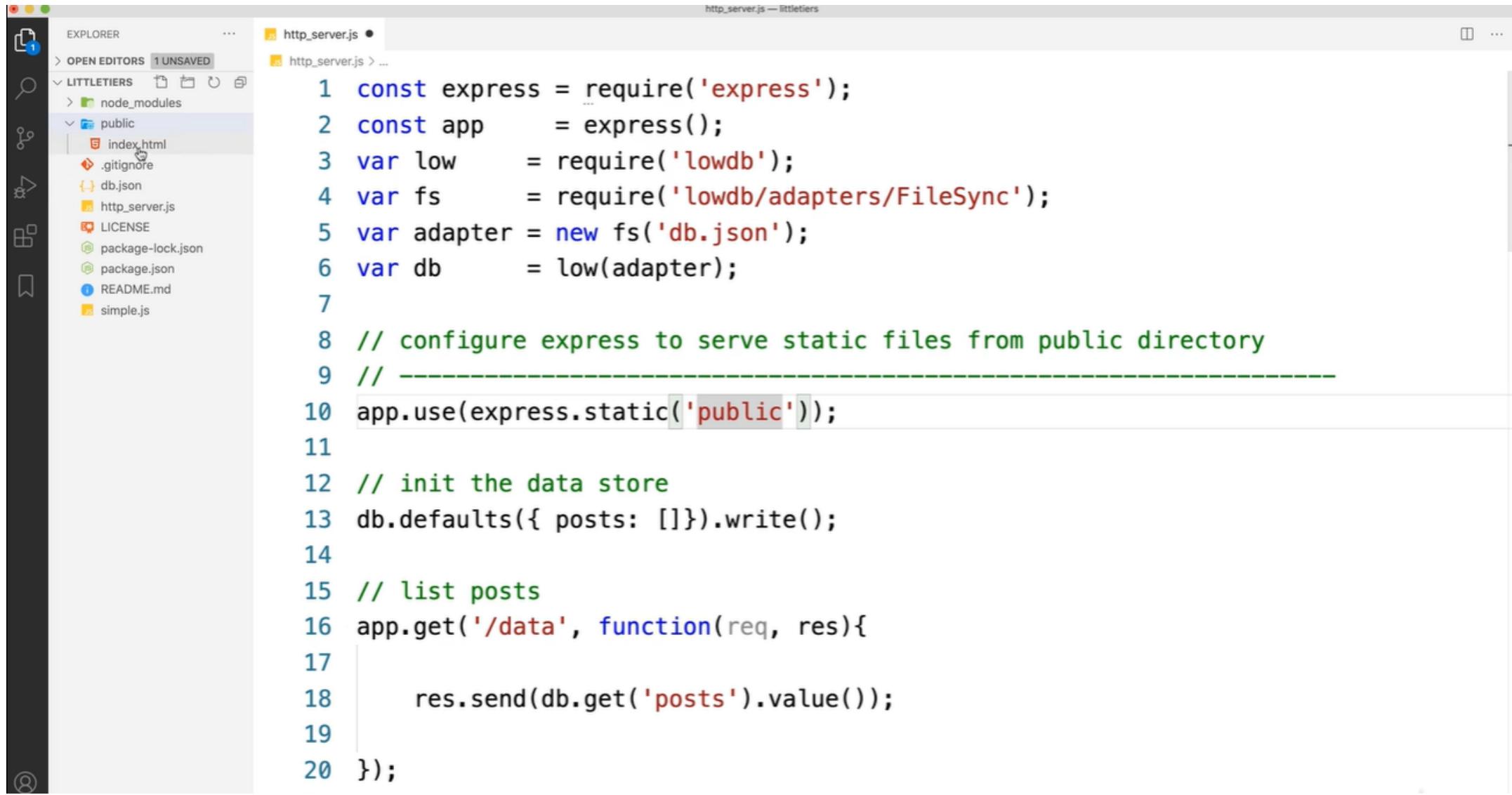
<https://getbootstrap.com/>

## HTTP Server – Serve Static HTML Files

```
// serve static files from “public” directory  
app.use(express.static('public'));
```



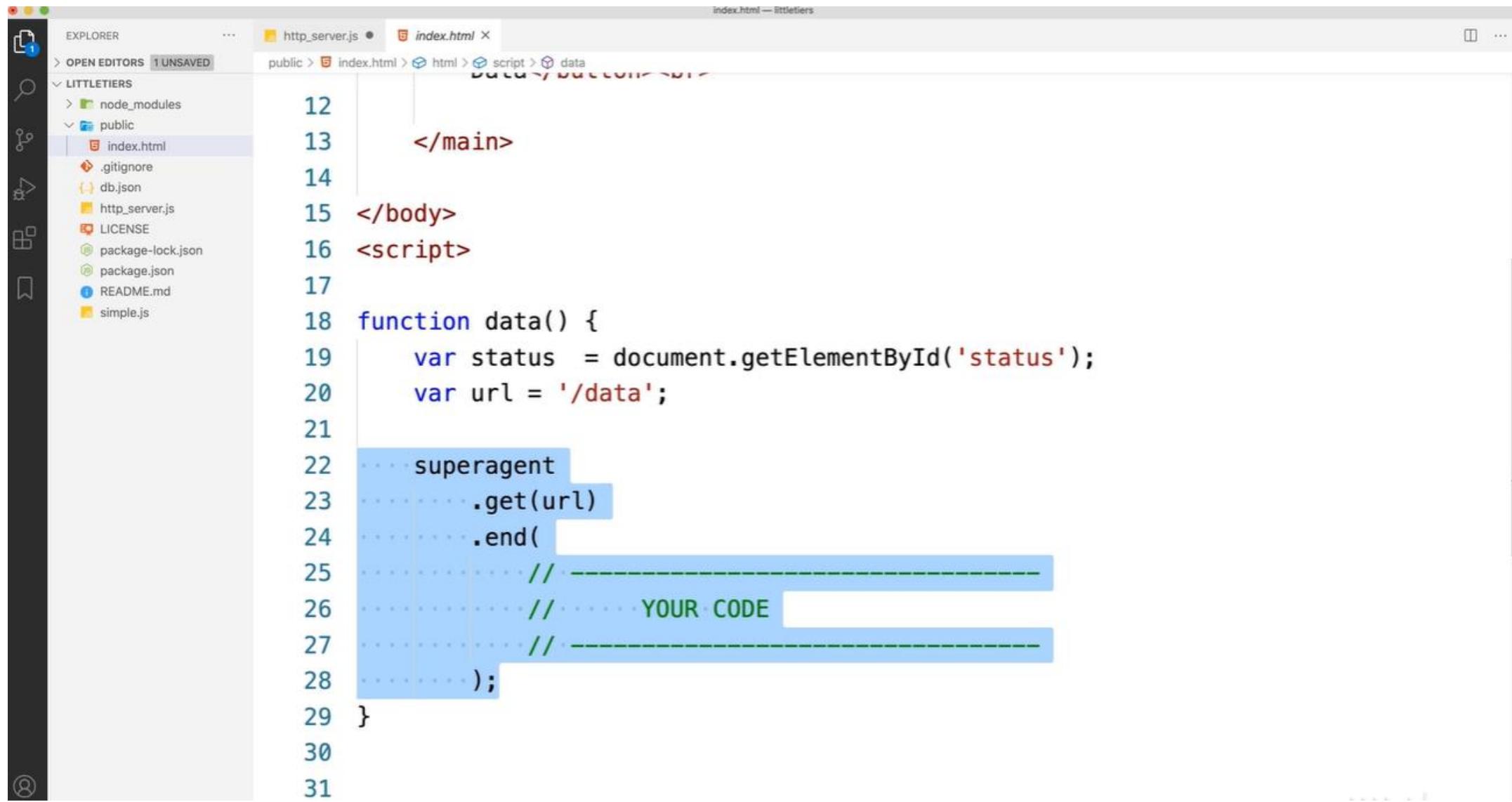
# Creating Three Tier Application (1/13)



The screenshot shows a Microsoft Visual Studio Code (VS Code) interface. On the left is the Explorer sidebar, which lists the project structure under 'LITTLETIERS'. The 'public' folder contains 'index.html', '.gitignore', 'db.json', 'http\_server.js', 'LICENSE', 'package-lock.json', 'package.json', 'README.md', and 'simple.js'. The main editor area displays the 'http\_server.js' file with the following code:

```
1 const express = require('express');
2 const app = express();
3 var low = require('lowdb');
4 var fs = require('lowdb/adapters/FileSync');
5 var adapter = new fs('db.json');
6 var db = low(adapter);
7
8 // configure express to serve static files from public directory
9 // -----
10 app.use(express.static('public'));
11
12 // init the data store
13 db.defaults({ posts: [] }).write();
14
15 // list posts
16 app.get('/data', function(req, res){
17
18     res.send(db.get('posts').value());
19 });
20 );
```

# Creating Three Tier Application (2/13)



The screenshot shows a code editor interface with the following details:

- Explorer View:** Shows a project structure under "LITTLETIERS". The "public" folder contains "index.html", ".gitignore", "db.json", "http\_server.js", "LICENSE", "package-lock.json", "package.json", "README.md", and "simple.js".
- Editor View:** The file "index.html" is open. The code is a template with some script logic.
- Code Content:**

```
12      </main>
13
14      </body>
15
16      <script>
17
18      function data() {
19          var status = document.getElementById('status');
20          var url = '/data';
21
22          superagent
23              .get(url)
24              .end(
25                  // ----- YOUR CODE -----
26                  // ----- YOUR CODE -----
27                  // ----- YOUR CODE -----
28              );
29      }
30
31
```

The code uses the `superagent` library to make an asynchronous GET request to the URL `/data`. The response is intended to be handled by the `YOUR CODE` placeholder.

# Creating Three Tier Application (3/13)

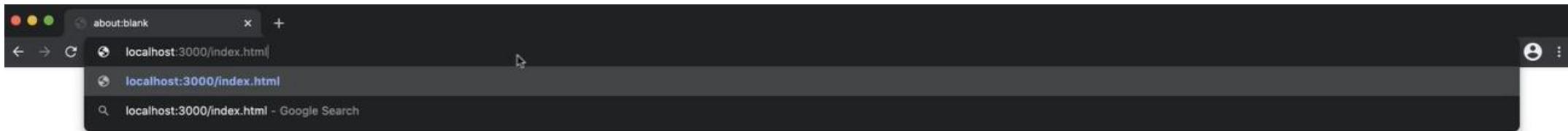
The screenshot shows a Microsoft Visual Studio Code (VS Code) interface. On the left is the Explorer sidebar with a tree view of the project structure. The main area has two tabs open: 'http\_server.js' and 'index.html'. The 'http\_server.js' tab contains the following code:

```
const express = require('express');
const app = express();
app.get('/', (req, res) => {
  res.send('Hello World!');
});
app.listen(3000, () => {
  console.log('Running on port 3000!');
});
```

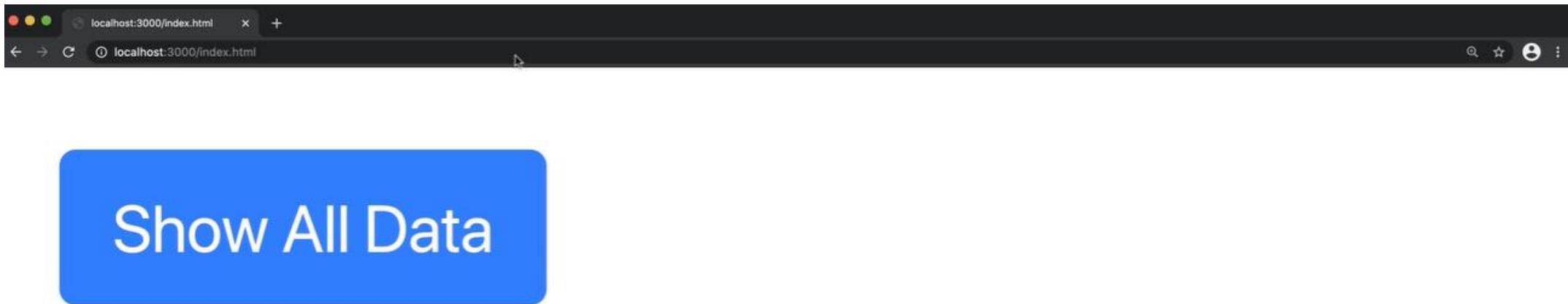
A terminal window is also visible, showing the command 'node http\_server.js' being run and the output 'Running on port 3000!'. The code editor window shows the continuation of the file with more code:

```
res.send(db.get('posts').value());
});
```

# Creating Three Tier Application (4/13)



## Creating Three Tier Application (5/13)

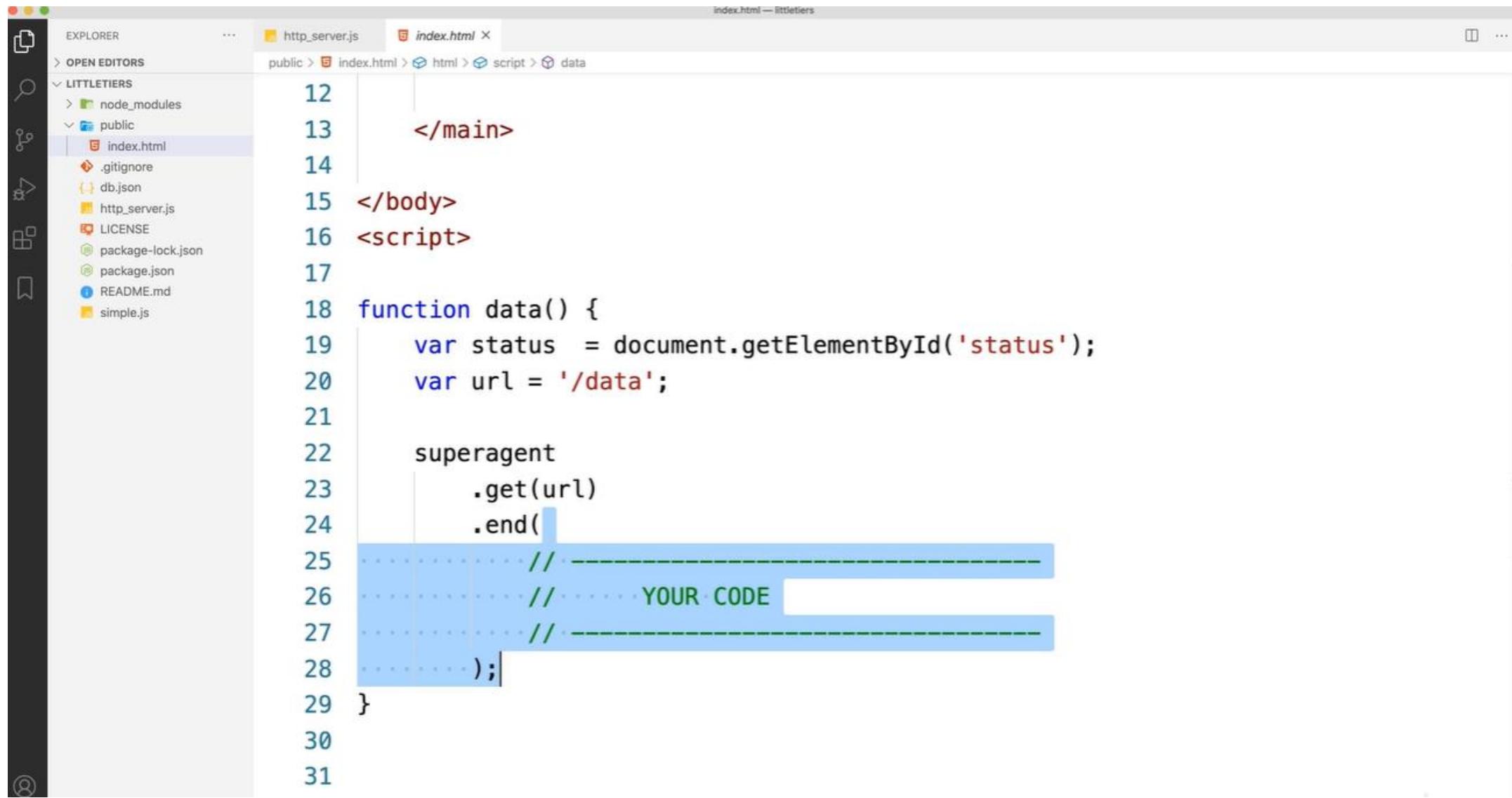


# Creating Three Tier Application (6/13)

A screenshot of a web browser window. The address bar shows 'localhost:3000/index.html' and 'view-source:localhost:3000/index.html'. The main content area displays the source code of 'index.html'. The code includes HTML structure with a main container and a status div, a button with an onclick event, and a script block containing a function named 'data()' which uses superagent to make a GET request to '/data'. There are three horizontal dashed lines with the text '-----' between them, followed by a note 'YOUR CODE'.

```
8 <main role="main" class="container" style="margin-top:20px">
9
10 <div id="status"></div>
11 <button type="button" class="btn btn-primary" onclick="data()>Show All Data</button><
12
13 </main>
14
15 </body>
16 <script>
17
18 function data() {
19     var status = document.getElementById('status');
20     var url = '/data';
21
22     superagent
23         .get(url)
24         .end(
25             // -----
26             //      YOUR CODE
27             // -----
28         );
29 }
30
31
```

# Creating Three Tier Application (7/13)

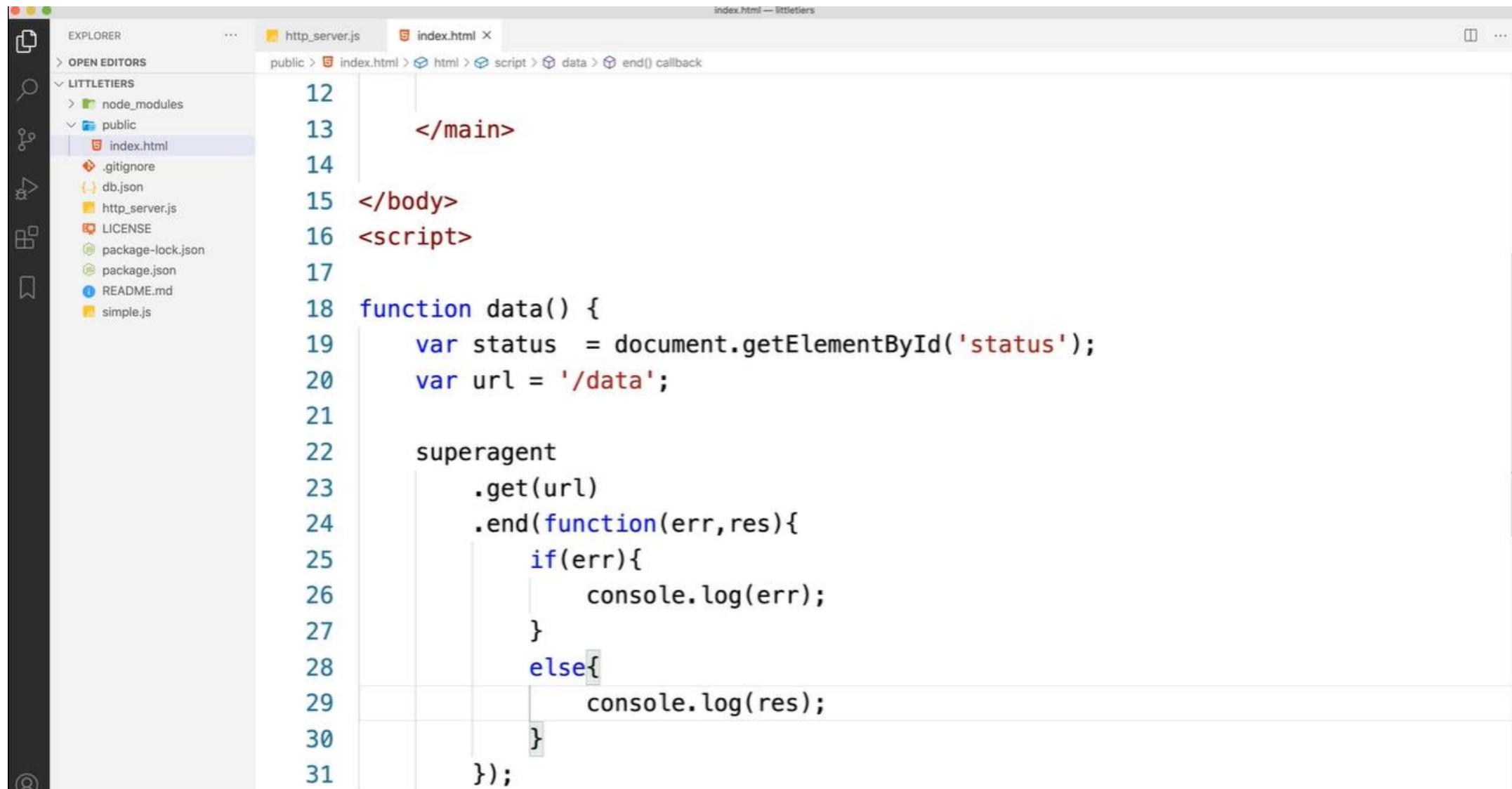


The screenshot shows a code editor interface with the file `index.html` open in the `littletiers` project. The code is a combination of HTML and JavaScript using the `superagent` library to fetch data from a URL.

```
12</main>
13
14</body>
15<script>
16
17
18function data() {
19    var status = document.getElementById('status');
20    var url = '/data';
21
22    superagent
23        .get(url)
24        .end(
25            // -----
26            // ... YOUR CODE ...
27            // -----
28        );
29}
```

The code includes placeholder comments for user code at lines 25, 26, and 27. The code editor has a light gray background with dark blue and red syntax highlighting. The sidebar on the left shows the project structure with files like `http_server.js`, `LICENSE`, and `simple.js`.

# Creating Three Tier Application (8/13)

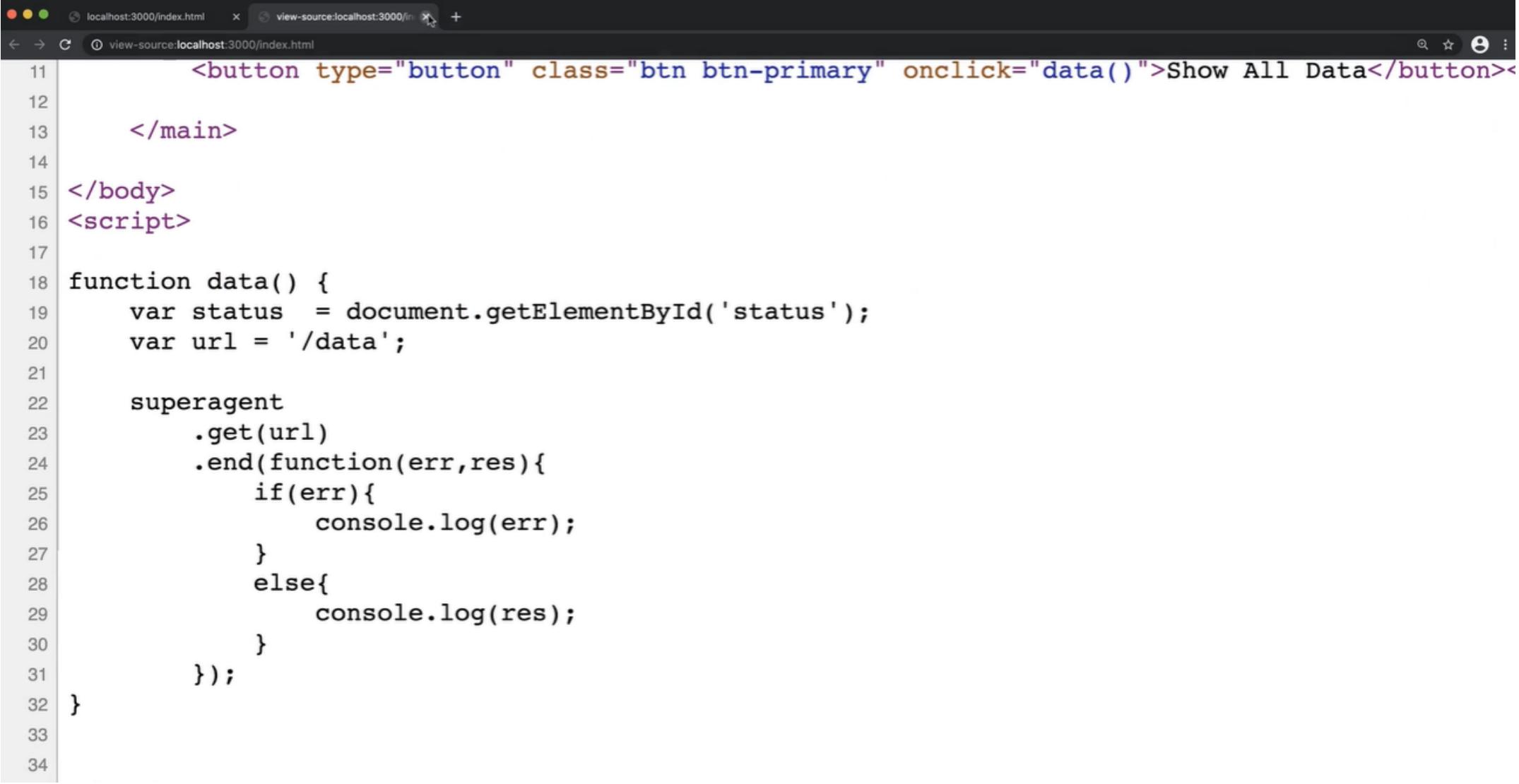


The screenshot shows a code editor interface with the file `index.html` open. The file contains the following code:

```
12 </main>
13
14 </body>
15 <script>
16
17
18 function data() {
19     var status = document.getElementById('status');
20     var url = '/data';
21
22     superagent
23         .get(url)
24         .end(function(err,res){
25             if(err){
26                 console.log(err);
27             }
28             else{
29                 console.log(res);
30             }
31         });
32 }
```

The code uses the `superagent` library to make an asynchronous GET request to the URL `/data`. It logs any errors to the console and logs the response to the console if no error occurs.

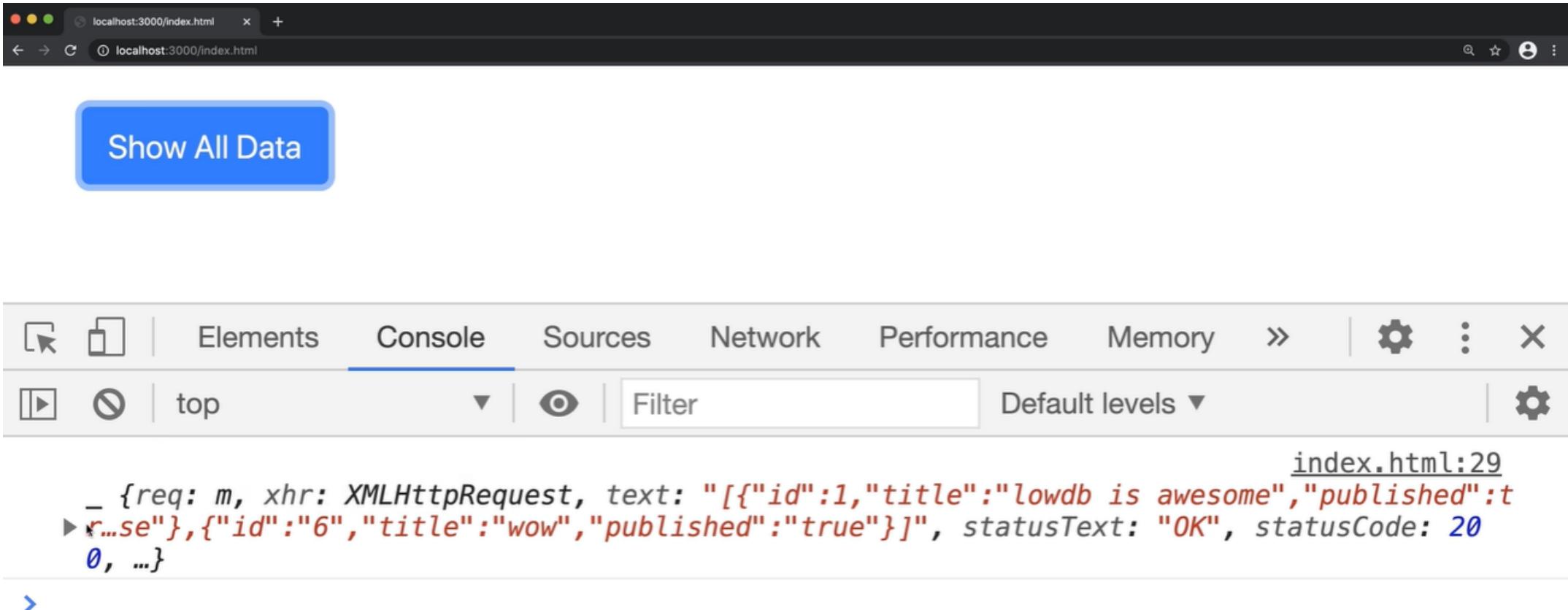
# Creating Three Tier Application (9/13)



The screenshot shows a browser window with two tabs: 'localhost:3000/index.html' and 'view-source:localhost:3000/index.html'. The source code is displayed in the second tab, showing the HTML structure and a JavaScript function named 'data()'.

```
11 <button type="button" class="btn btn-primary" onclick="data()">Show All Data</button>
12
13 </main>
14
15 </body>
16 <script>
17
18 function data() {
19     var status = document.getElementById('status');
20     var url = '/data';
21
22     superagent
23         .get(url)
24         .end(function(err,res){
25             if(err){
26                 console.log(err);
27             }
28             else{
29                 console.log(res);
30             }
31         });
32 }
33
34
```

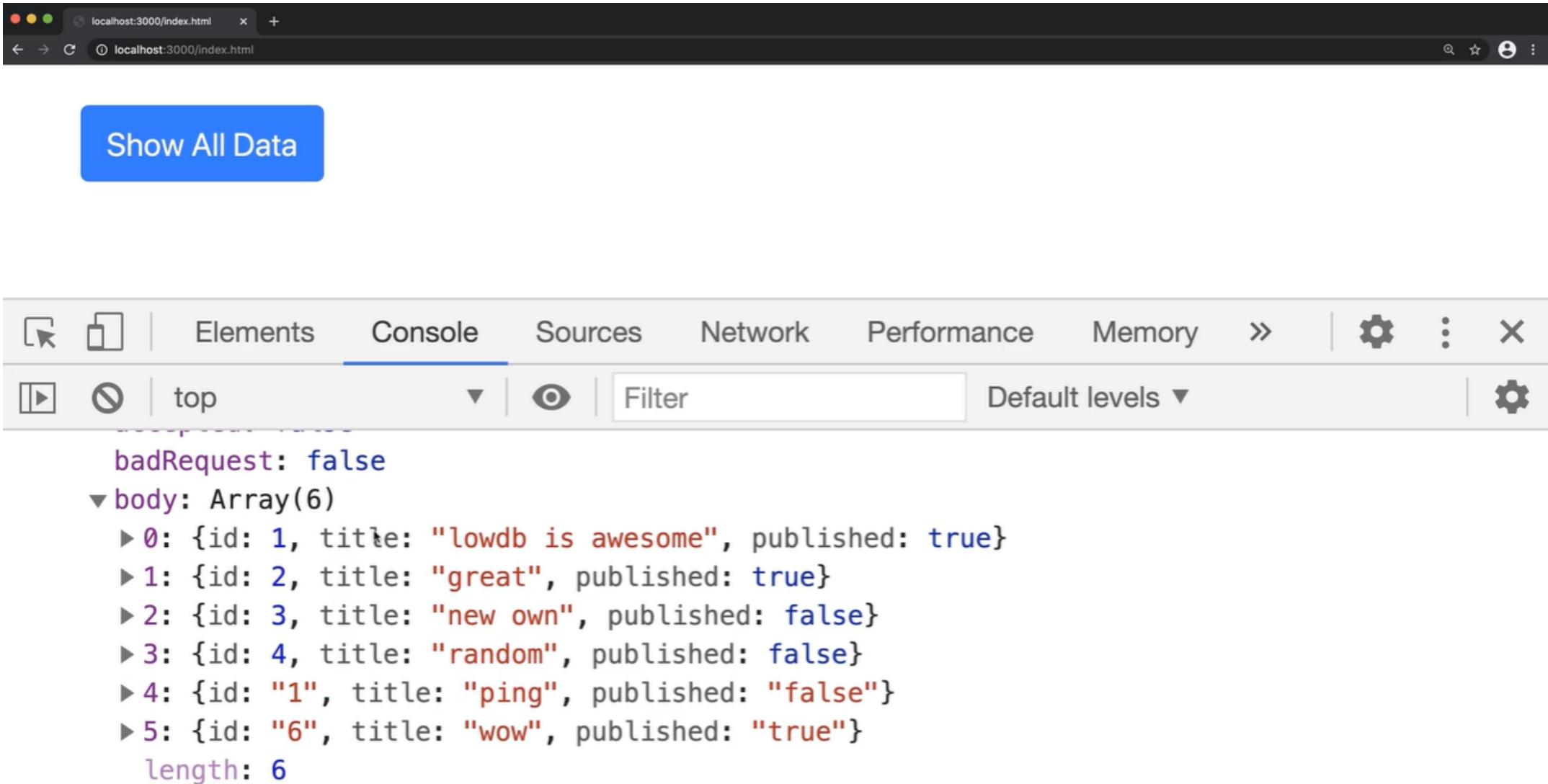
# Creating Three Tier Application (10/13)



The screenshot shows a web browser window with a dark theme. At the top, there's a header bar with a red, yellow, and green window control buttons, a title bar showing "localhost:3000/index.html", and a search bar with the same URL. Below the header is a blue button with white text that says "Show All Data". The main content area is mostly blank. At the bottom, a developer tools console is open, featuring tabs for Elements, Console, Sources, Network, Performance, and Memory. The "Console" tab is selected and highlighted in blue. The console interface includes icons for back, forward, and search, followed by the tab names. Below the tabs is a toolbar with play/pause, stop, and filter buttons, along with dropdown menus for "Default levels" and settings. The main content area of the console displays a JSON object. The object has a property "text" containing a multi-line string of JSON data. The string starts with "[{"id":1,"title":"lowdb is awesome","published":true}, {"id":6,"title":"wow","published":true}]". The line "published":true" is colored blue, while the rest of the string is in red. Above this JSON object, the file name "index.html:29" is displayed in blue. There are also some blue numbers and symbols at the bottom left of the console area.

```
index.html:29
{
  req: m, xhr: XMLHttpRequest, text: "[{"id":1,"title":"lowdb is awesome","published":true}, {"id":6,"title":"wow","published":true}]", statusText: "OK", statusCode: 200, ...
}
```

# Creating Three Tier Application (11/13)



The screenshot shows a web browser window with a dark theme. At the top, there is a navigation bar with a back arrow, forward arrow, refresh button, and a search bar containing the URL "localhost:3000/index.html". Below the navigation bar is a large blue button with white text that says "Show All Data".

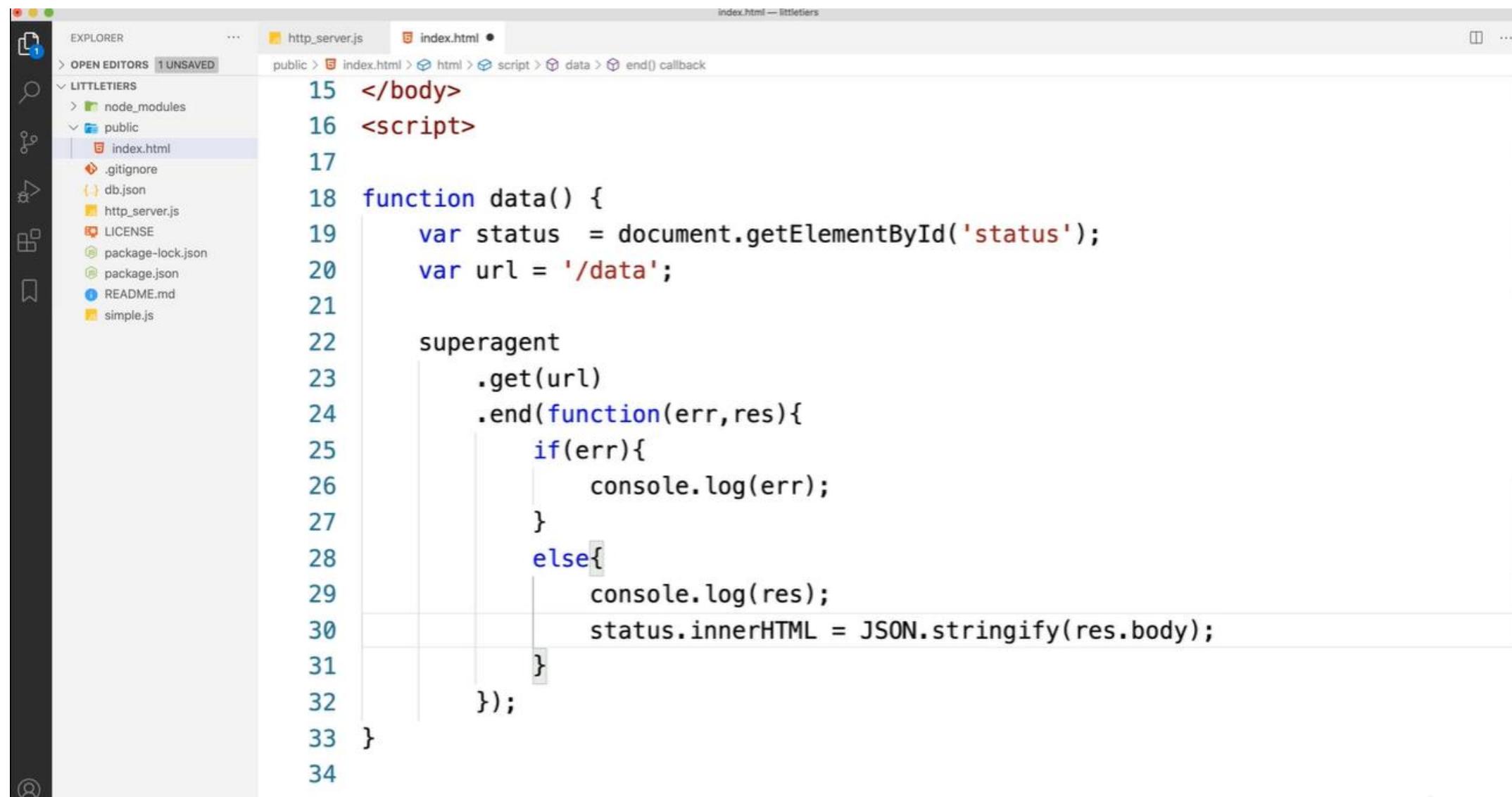
The main content area of the browser is currently empty, showing a plain white page.

At the bottom of the screen, an open developer tools console is visible. The tabs at the top of the console are "Elements", "Console" (which is selected and highlighted in blue), "Sources", "Network", "Performance", and "Memory". There are also icons for "More Tools" and "Close".

The "Console" tab contains the following output:

```
badRequest: false
▼ body: Array(6)
  ► 0: {id: 1, title: "lowdb is awesome", published: true}
  ► 1: {id: 2, title: "great", published: true}
  ► 2: {id: 3, title: "new own", published: false}
  ► 3: {id: 4, title: "random", published: false}
  ► 4: {id: "1", title: "ping", published: "false"}
  ► 5: {id: "6", title: "wow", published: "true"}
  length: 6
```

# Creating Three Tier Application (12/13)

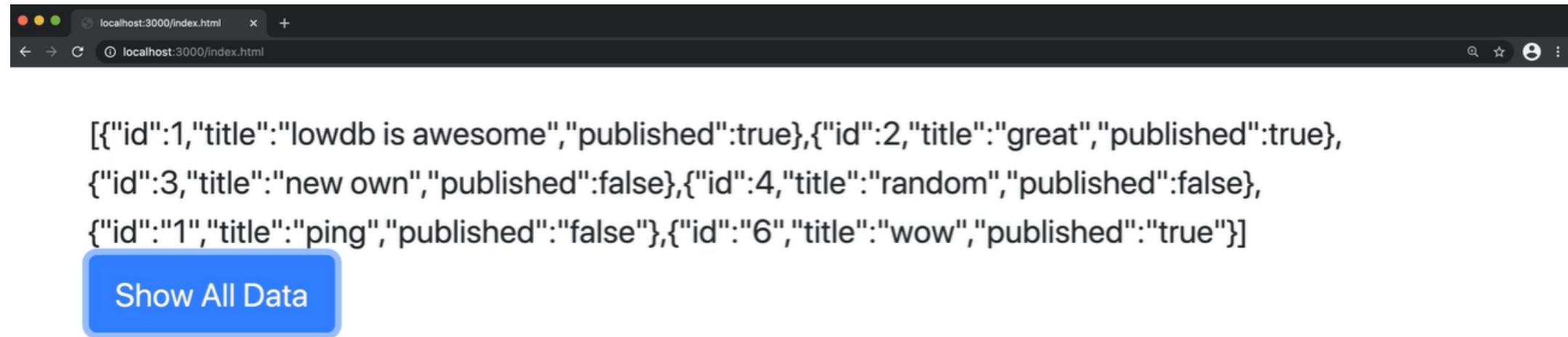


The screenshot shows a code editor interface with the following details:

- Explorer View:** Shows a file tree for a project named "LITTLETIERS". The "public" folder contains "index.html", which is currently selected.
- Editor View:** Displays the content of "index.html". The code uses the Superagent library to make an asynchronous GET request to "/data". It logs errors to the console and updates the "status" element's innerHTML with the JSON response body.

```
15  </body>
16  <script>
17
18  function data() {
19      var status = document.getElementById('status');
20      var url = '/data';
21
22      superagent
23          .get(url)
24          .end(function(err,res){
25              if(err){
26                  console.log(err);
27              }
28              else{
29                  console.log(res);
30                  status.innerHTML = JSON.stringify(res.body);
31              }
32          });
33 }
34
```

# Creating Three Tier Application (13/13)



# Receiving Data From The User Interface (1/)

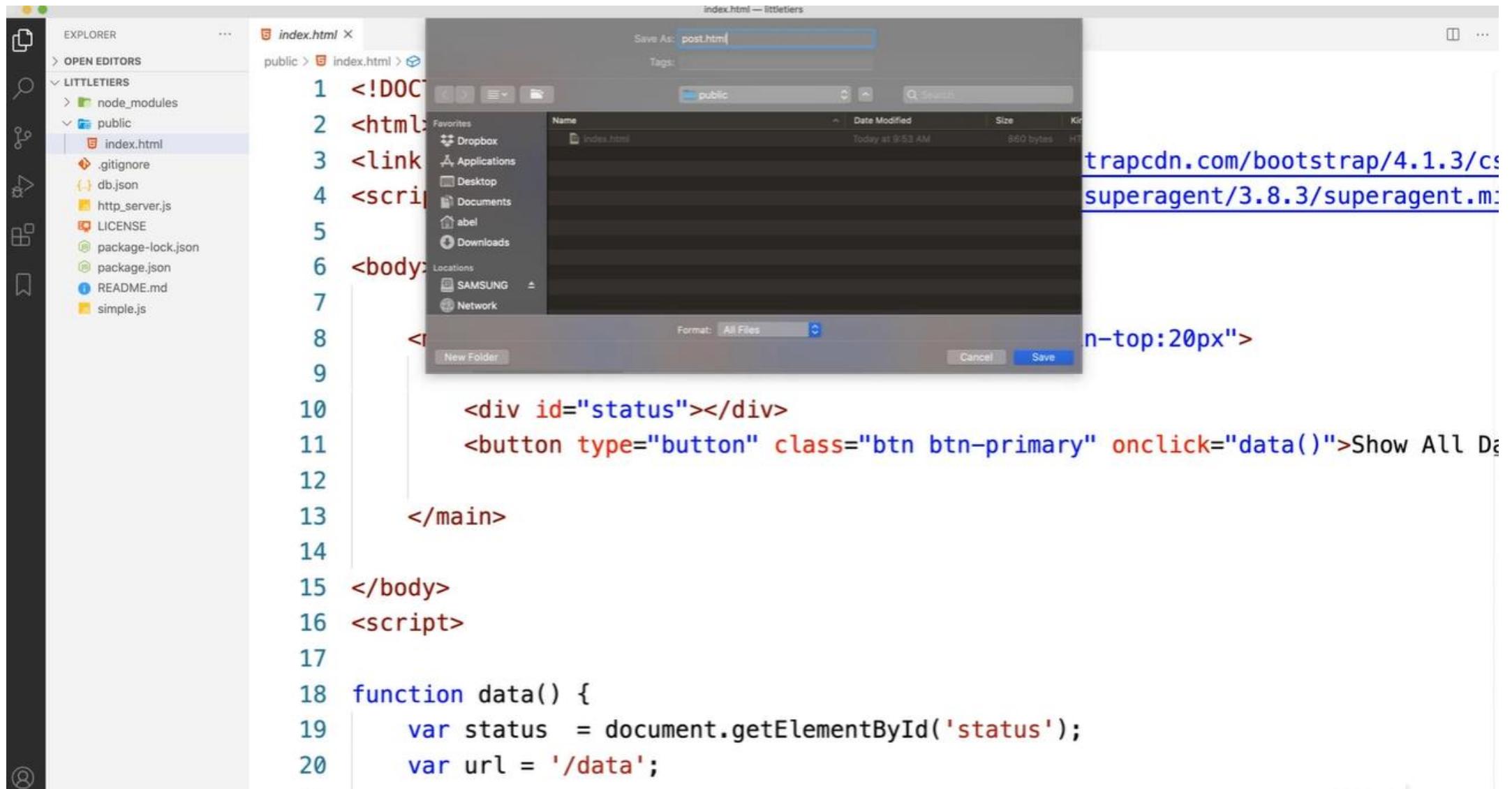


A screenshot of a web browser window titled "localhost:3000/post.html". The window contains a form with three input fields: "id", "title", and "published", each in its own row. Below the input fields is a blue "submit" button.

id
title
published

submit

# Receiving Data From The User Interface (2/)



The screenshot shows a code editor interface with the following details:

- EXPLORER** sidebar on the left showing project structure:
  - LITTLETIERS
  - public
  - index.html
  - .gitignore
  - db.json
  - http\_server.js
  - LICENSE
  - package-lock.json
  - package.json
  - README.md
  - simple.js
- index.html** editor tab showing the following code:

```
1 <!DOCTYPE html>
2 <html>
3   <link href="https://trapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css" rel="stylesheet">
4   <script src="https://superagent/3.8.3/superagent.min.js" type="text/javascript">
5
6   <body>
7     <div id="status" style="border: 1px solid black; padding: 10px; margin-top: 20px">
8       <div id="content"></div>
9
10      <button type="button" class="btn btn-primary" onclick="data()">Show All Data</button>
11
12    </div>
13  </main>
14
15 </body>
16 <script>
17
18 function data() {
19   var status = document.getElementById('status');
20   var url = '/data';
```
- Save As** dialog box in the center showing "post.html" in the "Name" field.
- File Explorer** sidebar on the right showing files in the "public" folder.



© MIT xPRO 2021. All rights reserved.