

NodeJS

פתרונות לתרגילים מש"ב

HTTP Built in Module

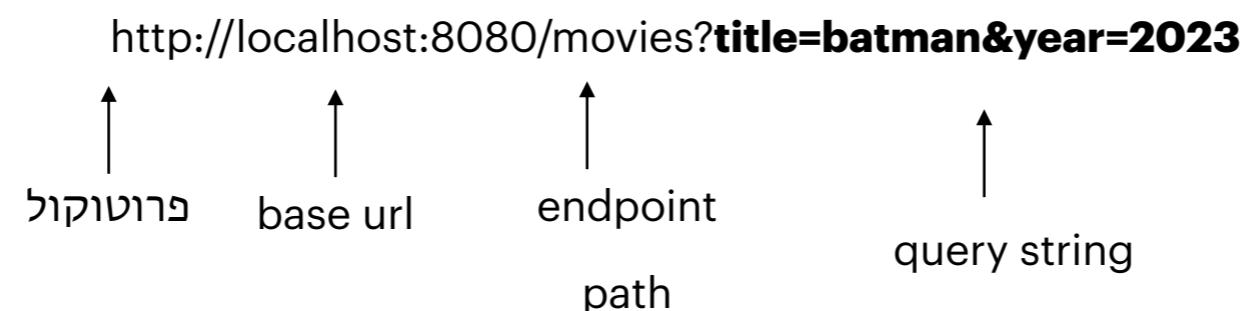
FS Built in module

Express

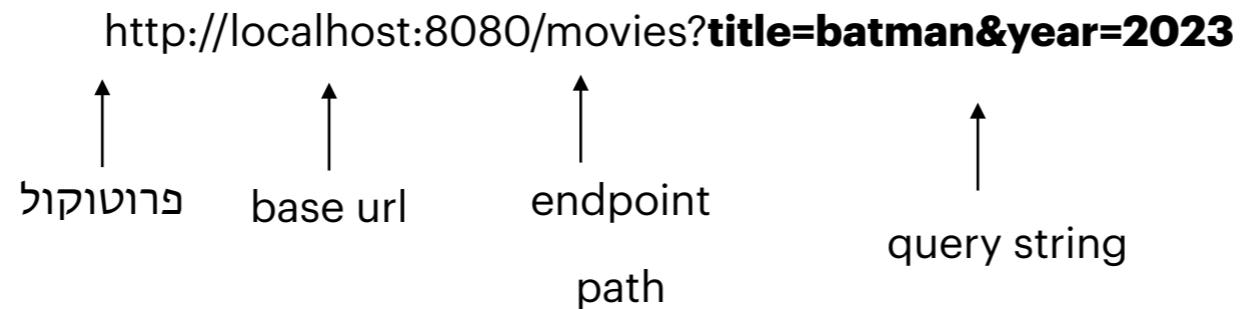
Middleware



שרשרת של פונקציות שmagicות לבקשת



פירוק של ה Query String מה URL



middleware

break-url.js

```
const url = require('node:url');

const breakUrl = (req, res) => {
  //req.url = /movies?title=batman
  const result = url.parse(req.url, true)

  console.dir(result.query);
  console.dir(result.pathname);
}

module.exports = breakUrl;
```

index.js

```
const breakUrl = require('./middleware/break-url');
const logger = require('./middleware/logger');
const router = require('./routes/router');
const http = require('http');

const server = http.createServer();

server.on('request', breakUrl);
server.on('request', logger);
server.on('request', router);

server.listen(8080);
```

ברגע אנחנו רק מדפסים ל `console`
נמשיך מכאן בשיעור הבא.

שיעור בית:

- בעזרת ממשק ה - cmd (או הטרמינל של מק) פתח פרויקט חדש בשם node-test ופתח אותו בכתבן (ide) של code vs
- צור בפרויקט קובץ package.json (בעזרת npm)

צור קובץ חדש בפרויקט בשם index.js ובו פקודה להציג בקונסול (console.log) מחרוזת תווים לבחירתך
- הרץ את הקובץ בעזרת ממשק ה - cmd (דרך cmd)

- צור קובץ חדש בשם my-module2.js עבור מודול משלך.
- במודול זה יצא פונקציה בשם demo שתחזיר מחרוזת תווים כל שהוא וכן משתנה נוסף בשם num שיכיל מספר כל שהוא לבחירתך
- עשה שימוש במודול זה בדף הראשי (index.js) שלך והציג בקונסול את הערך המוחזר מהפונקציה demo וכן את ערכו של המשתנה num

צור קובץ חדש בשם product.js ומחלקה בשם Product, הגדר במחלקה זו מאפיין בשם price שיכיל ערך כל שהוא לבחירתך וכן מתודה בשם getPrice שתחזיר את ערכו של המאפיין price
- ייצא מודול זה (product) ע"י שימוש ב - (module.exports)
- בקובץ index.js ייבא את המחלקה Product והציג בקונסול את הערך המוחזר מהמתודה getPrice

היעזר במודול מובנה של node בשם os (קישור של operating system) כדי להציג בקונסול כמה מקום פניו יש מחשב שלך מביתנית זיכרון מערכת
(חפשו בדוקומנטציה את ה- module built in os ופלו בהתאם לדוקומנטציה).

משימת
חיפוש

היעזר במודול path של node כדי להציג בקונסול רק את סימות הקובץ שממנו אתה מרץ את הקובץ (index.js לדוגמה)
(חפשו בדוקומנטציה את ה- module built in path ופלו בהתאם לדוקומנטציה).

משימת
חיפוש

נתון המערך הבא: [1, 2, 1, 4, 1, 3], יש להציג רק את הערכים הייחודיים במערך. כתבו פעולה במודול utils שתחזיר רק את הערכים הייחודיים במערך.

משימת
מעניינת

נתון המערך הבא: [1, 2, 1, 4, 1, 3], יש להציג רק את הערכים הייחודיים במערך. תוך שימוש בספרייה underscore תוך שימוש בספרייה underscore יש לקרוא בדוקומנטציה של הספרייה לגבי הפעולה `uniq` ולהשתמש בה.

משימת
חיפוש

שיעור בית - HTTP:

צרו אובייקט של שרת יאזין בפורט 5500

השרת יאזין לבקשתים בכתובות הבאות:

localhost:5000/about

localhost:5000/home

localhost:5000/frontend

localhost:5000/backend

localhost:5000/fullstack

בכל בקשה לאחד הנתיבים הללו בדף יש להחזיר מחרוזת מתאימה.

אם הלקוח מנסה לנוט לדף שלא הוגדר יש להחזיר בתגובה

.404 - not found

*צרו קובץ בשם router וממשו בו את הלוגיקה של ניתוב הבקשות.

הוסיפו logger

שמדפיס את כל הבקשות שנכנסות.

בפתרון יש להשתמש ב[http module](#) המובנה של Node.js

פתרונות לתרגילים נבחרים:

נתון המערך הבא: [1, 2, 1, 4, 1, 3]. יש להציג רק את הערכים הייחודיים במערך. - כתבו פעולה במודול `utils` שתחזיר רק את הערכים הייחודיים במערך.

משמעות
משמעות

ניצור אובייקט ריק:

```
const obj = {}
```

עדכן את האובייקט כך שהמפתחות יכילו את האלמנטים במערך:

והערכים - `counter` שਮונה כמה פעמים האלמנט הופיע.

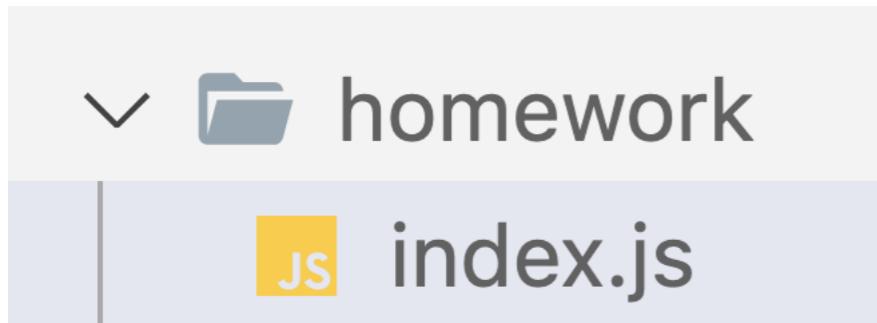
```
const obj = {"1": 3, "2":1, "3":1, "4":1}
```

בסיום - נróż על האובייקט ונציג את המפתחות שלו כפתרונות.

פתרונות לתרגילים נבחרים:

נתון המערך הבא: [1, 2, 1, 4, 1, 3]. יש להציג רק את הערכים הייחודיים במערך. - כתבו פוליה במודול `utils` שתחזיר רק את הערכים הייחודיים במערך.

משימה
מענית



תיקיה חדשה
קובץ חדש - homework/index.js

A screenshot of a terminal window showing the command `index.js .` being run. The terminal also shows the path `homework > index.js > <unknown>`.

```
1 const hw = () => {  
2  
3 }  
4  
5 module.exports = { hw }
```

פתרונות לתרגילים נבחרים:

נתון המערך הבא: [3,1,4,1,2,1]. יש להציג רק את הערכים הייחודיים במערך. - כתבו פעולה במודול utils שתחזיר רק את הערכים הייחודיים במערך.

משימה
מעניינת

```
const hw = (arr = [1, 2, 1, 4, 1, 3]) => {
    const obj = {};

    //{"1": 0, "2":0, "3":0, "4":0}
    for (let item of arr) {
        obj[`$ {item}`] = 0;
    }

    //obj['1'] = 0;
    //obj['2'] = 0;
    //obj['1'] = 0;
    //obj['4'] = 0;
    //obj['1'] = 0;
    //obj['3'] = 0;
    //{"1": 0, "2":0, "3":0, "4":0}
}

module.exports = { hw }

//hw() (arr=[])
//hw([1, 1, 1, 1, 2, 3])
```

איצץיה על מפתחות של אובייקט:

```
const p1 = {  
    "firstName": "dave",  
    "lastName": "green"  
}  
  
for(const k in p1){  
    console.log(k); // "firstName", "lastName"  
}
```

```
const p1 = {  
    "firstName": "dave",  
    "lastName": "green"  
}  
  
console.log(p1.firstName); // "dave"  
console.log(p1['firstName']); // "dave"
```

דרך נוספת לגישה לערך באובייקט באמצעות המפתח

פתרונות לתרגילים נבחרים:

נתון המערך הבא: [3,1,2,1,4,1]. יש להציג רק את הערכים הייחודיים במערך. - כתבו פוליה במודול utils שתחזיר רק את הערכים הייחודיים במערך.

משימה
מעניינת

```
const hw = (arr = [1, 2, 1, 4, 1, 3]) => {
    const obj = {};

    //{"1": 0, "2":0, "3":0, "4":0}
    for (let item of arr) {
        obj[`$ {item}`] = 0;
    }

    const distinct = [];
    //loop over the object keys: "1", "2", "3", "4"
    for (let k in obj) {
        distinct.push(Number(k))
    }
    return distinct;
}

module.exports = { hw }

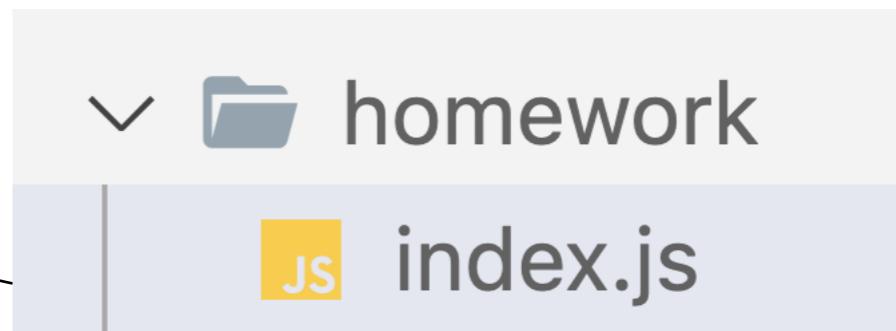
//hw() (arr=[])
//hw([1, 1, 1, 1, 2, 3])
```

פתרונות לתרגילים נבחרים:

נתון המערך הבא: [1, 2, 1, 4, 1, 3]. יש להציג רק את הערכים הייחודיים במערך. - כתבו פעולה במודול `utils` שתחזיר רק את הערכים הייחודיים במערך.

משמעות
משימה

```
const { hw } = require('./homework');
console.log(hw());
```



כשמייבאים קובץ index מתיקיה -
לא חייבים לרשום index

זה מיובה אוטומטית.

פתרונות לתרגילים נבחרים: הרחבה לפתרון - שאלות שנשאלו:

משמעות
משימה

נתון המערך הבא: [1, 2, 1, 4, 1, 3]. יש להציג רק את הערכים הייחודיים במערך. כתבו פעולה במודול utils שתחזיר רק את הערכים הייחודיים במערך.

```
const hw = (arr = [1, 2, 1, 4, 1, 3]) => {
  const obj = {};

  //{"1": 0, "2":0, "3":0, "4":0}
  for (let item of arr) {
    obj[`#${item}`] = 0;
  }
}
```

```
for (let item of arr) {
  obj[`#${item}`] += 1;
}
```

בכל איטרציה על ערך במערך:

נקדם את counter באובייקט

(מקדמים את counter בכל פעם שפוגשים בערך)

```
const distinct = [];
//loop over the object keys: "1", "2", "3", "4"
for (let k in obj) {
  const count = obj[k];
  if (count === 1) { ←
    distinct.push(Number(k))
  }
}
return distinct;

module.exports = { hw }
```

בלולה الأخيرة:

נחלץ גם את הערך המספרי של הcounter וرك אם counter הוא 1 נכניס אותו למערך התשובות.

פתרו באמצעות Set

נתון המערך הבא: [1,2,1,4,1,3], יש להציג רק את הערכים הייחודיים במערך. - כתבו פוליה במודול utils שתחזיר רק את הערכים הייחודיים במערך.

```
const hwv2 = (arr = [1, 2, 1, 4, 1, 3]) => {  
    const myFirstSet = new Set(arr)  
    return [...myFirstSet];  
}  
  
module.exports = { hw, hwv2 }
```

מבנה נתונים Set

מבנה נתונים שמנוע כפילות - אוטומטית.
מסנן את הכפולים - משאיר רק ערכים unique

מבנה נתונים מערך
מבנה נתונים - אובייקט
מבנה נתונים - Set
מנוע כפילות

פתרונות לתרגילים נבחרים:

שימוש בספריה:

נתון המערך הבא: [1,2,1,4,1,3], יש להציג רק את הערכים הייחודיים במערך. תוך שימוש בספריה underscore יש לקרוא בדוקומנטציה של הספריה לגבי הפעולה `uniq` ולהשתמש בה.

משימת
חיפוש

<https://underscorejs.org/#uniq>

npm i underscore

uniq `_.uniq(array, [isSorted], [iteratee])` Alias: **unique** [SOURCE](#)
Produces a duplicate-free version of the **array**, using `==` to test object equality. In particular only the first occurrence of each value is kept. If you know in advance that the **array** is sorted, passing `true` for **isSorted** will run a much faster algorithm. If you want to compute unique items based on a transformation, pass an **iteratee** function.

`_.uniq([1, 2, 1, 4, 1, 3]);`
`=> [1, 2, 4, 3]`

```
const { uniq } = require('underscore');

const hwv3 = (arr = [1, 2, 1, 4, 1, 3]) => {
  return uniq(arr);
}

module.exports = { hw, hwv2, hwv3 }
```

פירוק של url - פיצול של ה Query String

```
> const url = "/movies?title=batman"
< undefined
> url.split("?")
< ▶ (2) ['/movies', 'title=batman']
```

```
const breakUrl = (req, res) => {
  //req.url = /movies?title=batman
  const split = req.url.split("?")
  const pathName = split[0];
  const query = split[1];

}
```

מודול מובנה:

```
const url = require('node:url');

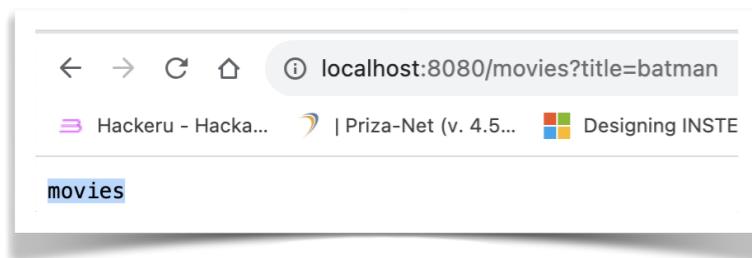
const breakUrl = (req, res) => {
  const result = url.parse(req.url, true);
  const pathName = result.pathname;
  const query = result.query;
}
```

במקום להגדיר משתנים
נשתמש ב **object destructuring**

```
const breakUrl = (req, res) => {
  const { pathname, query} = url.parse(req.url, true);
}
```

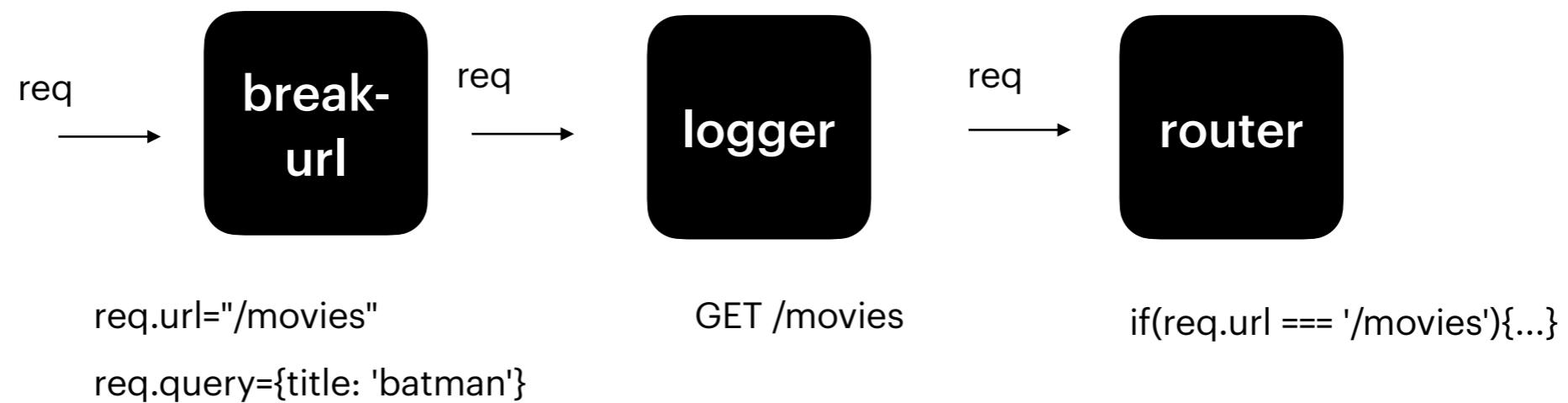
Middleware

שרשרת של פונקציות לטיפול בבקשת:



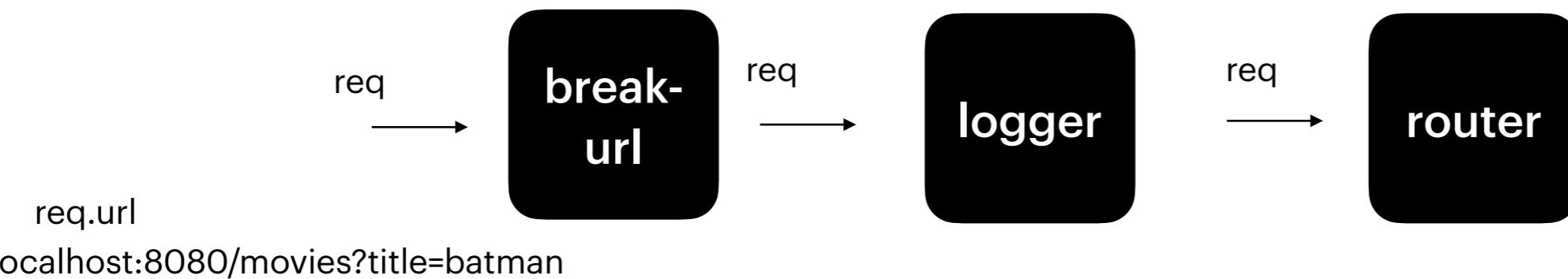
req.url

localhost:8080/**movies?title=batman**



Middleware

שרשרת של פונקציות לטיפול בבקשת:



מותר לפונקציה `break-url`
לשנות url ולהוסיף ערכים לreq

req.url = pathname

req.query = query

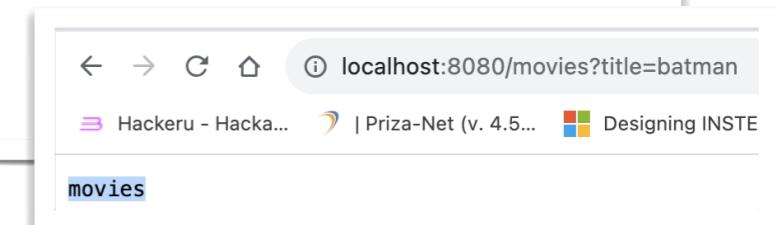
```
const url = require('node:url');

const breakUrl = (req, res) => {
  //movies?title=batman
  const { pathname, query } = url.parse(req.url, true);

  req.url = pathname; //req.url = 'movies'
  req.query = query; //req.query = {title: 'batman'}

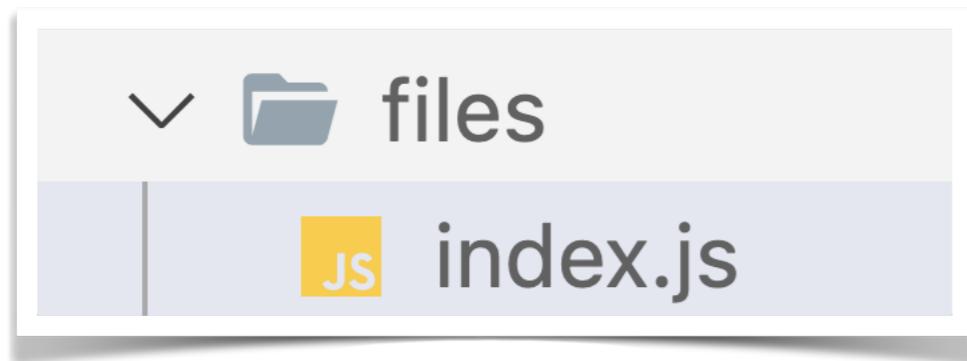
}

module.exports = breakUrl;
```



עבודה עם קבצים ב-node

מודול מובנה: fs



תיקיה חדשה
קובץ index.js

יצירת תיקיה:

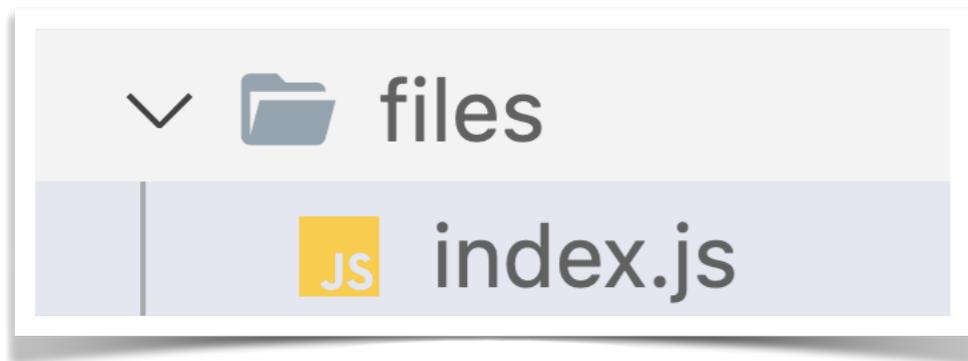
<https://nodejs.org/api/fs.html#fsmkdirpath-options-callback>

```
import { mkdir } from 'node:fs';

// Create ./tmp/a/apple, regardless of whether ./tmp and ./tmp/a exist.
mkdir('./tmp/a/apple', { recursive: true }, (err) => {
  if (err) throw err;
});
```

עבודה עם קבצים ב-node

מודול מובנה: fs



תיקיה חדשה
קובץ index.js

```
const { mkdir } = require('node:fs')

const demoCreateDir = () => {
  mkdir("demo", (err) => {
    if (err) {
      console.error(err);
    } else {
      console.log("Dir demo created")
    }
  });
}

module.exports = { demoCreateDir }
```

הפעלה של הפקציה:

```
const { demoCreateDir } = require('./files');

demoCreateDir();
```

עבודה עם קבצים ב-node

יצירת תיקיה עם תת-תיקיות:

```
const { mkdir } = require('node:fs')

const demoCreateDir = () => {
  mkdir("usr/images/icons", { recursive: true }, (err) => {
    if (err) {
      console.error(err);
    } else {
      console.log("Dir demo created")
    }
  });
}

module.exports = { demoCreateDir }
```

עבודה עם קבצים ב-node

יצירת תיקיה עם תת תיקיות:

```
const { mkdir } = require('node:fs')

const demoCreateDirRecursive = () => {
  mkdir("usr/images/icons", { recursive: true }, (err) => {
    if (err) {
      console.error(err);
    } else {
      console.log("Dir usr/images/icons created")
    }
  });
}

const demoCreateDir = () => {
  mkdir("demo", (err) => {
    if (err) {
      console.error(err);
    } else {
      console.log("Dir demo created")
    }
  });
}

module.exports = { demoCreateDir }
```

יצירת תיקיה עם תת תיקיות:

יצירת תיקיה רגילה:

כתיבה לקובץ חדש - writeFile

```
import { writeFile } from 'node:fs';
import { Buffer } from 'node:buffer';

const data = new Uint8Array(Buffer.from('Hello Node.js'));
writeFile('message.txt', data, (err) => {
  if (err) throw err;
  console.log('The file has been saved!');
});
```

Example 1:



```
// Node.js program to demonstrate the
// fs.writeFile() method

// Import the filesystem module
const fs = require('fs');

let data = "This is a file containing a collection of books.";

fs.writeFile("books.txt", data, (err) => {
  if (err)
    console.log(err);
  else {
    console.log("File written successfully\n");
    console.log("The written has the following contents:");
    console.log(fs.readFileSync("books.txt", "utf8"));
  }
});
```

כתיבה לקובץ חדש - writeFile

files/index.js

```
const demoCreateFile = () => {
  writeFile('demo/hello.txt', "Hello, World!", {}, (err) => {
    if (err) {
      console.error(err);
    } else {
      console.log("file created")
    }
  })
}

module.exports = {demoCreateFile, demoCreateDir}
```

index.js

```
const { demoCreateDir, demoCreateFile } = require('./files');
demoCreateFile();
```

מיני תרגיל:

חפשו בדוקומנטציה איך לקרוא מקובץ רמז



התוצאה - מחרוזת עם התוכן של הקובץ

Hello, World!

אחרי התרגיל - הסבר על UTF

מינি תרגיל:

חפשו בדוקומנטציה איך לקרוא מקובץ רמז

```
//const {readFile} = require('node:fs')
const demoReadFile = () => {
  readFile('demo/hello.txt', (err, data) => {
    console.log(data);
  });
}
```

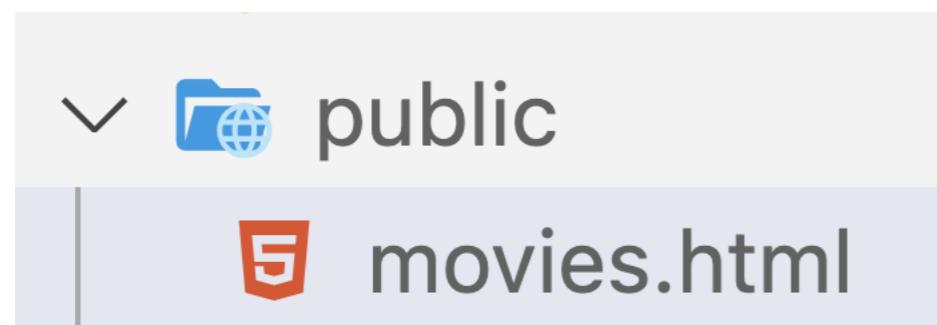
מדפיס את המידע הבינארי בקובץ:

```
//const {readFile} = require('node:fs')
const demoReadFile = () => {
  readFile('demo/hello.txt', { encoding: 'utf-8' }, (err, data) => {
    console.log(data);
  });
}
```

מדפיס את המידע הבינארי בקובץ כתקסט לפי קידוד של utf-8:

כדי להוסיף התייחסות לבדיקה שגיאה.

הגשה של קבצים: שרת http יכול להגיש גם קבצים.



movies.html

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>movies</title>
</head>

<body>
    <h1>Movies</h1>
</body>

</html>
```

הגשה של קבצים:

routes/router.js

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

const router = (req, res) => {
  if (req.url === '/home') {
    return res.end("Home");
  }
  if (req.url === '/about') {
    return res.end("about");
  }
  if (req.url === '/movies') {
    fs.readFile("public/movies.html", (err, data) => {
      res.end(data);
    })
    return
  }
  res.end("Not Found");
}

module.exports = router;
```

בקשה ל.movies

מחזיר את התוכן של הקובץ response

הספריה :Express

1 ניצור תיקיה חדשה לפרויקט חדש:

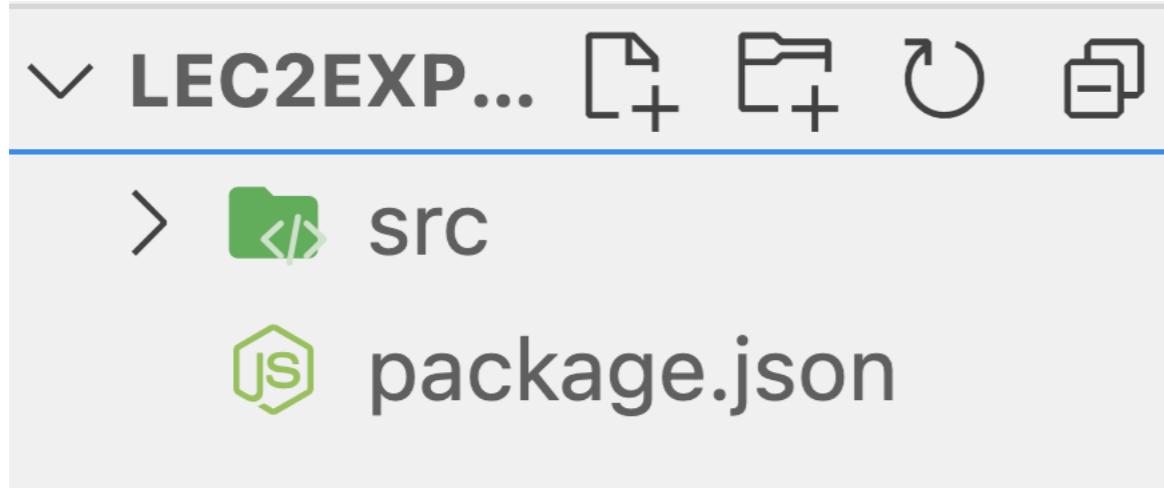


2 נגרור את התיקיה לvscode

file->open folder->Lec2Express

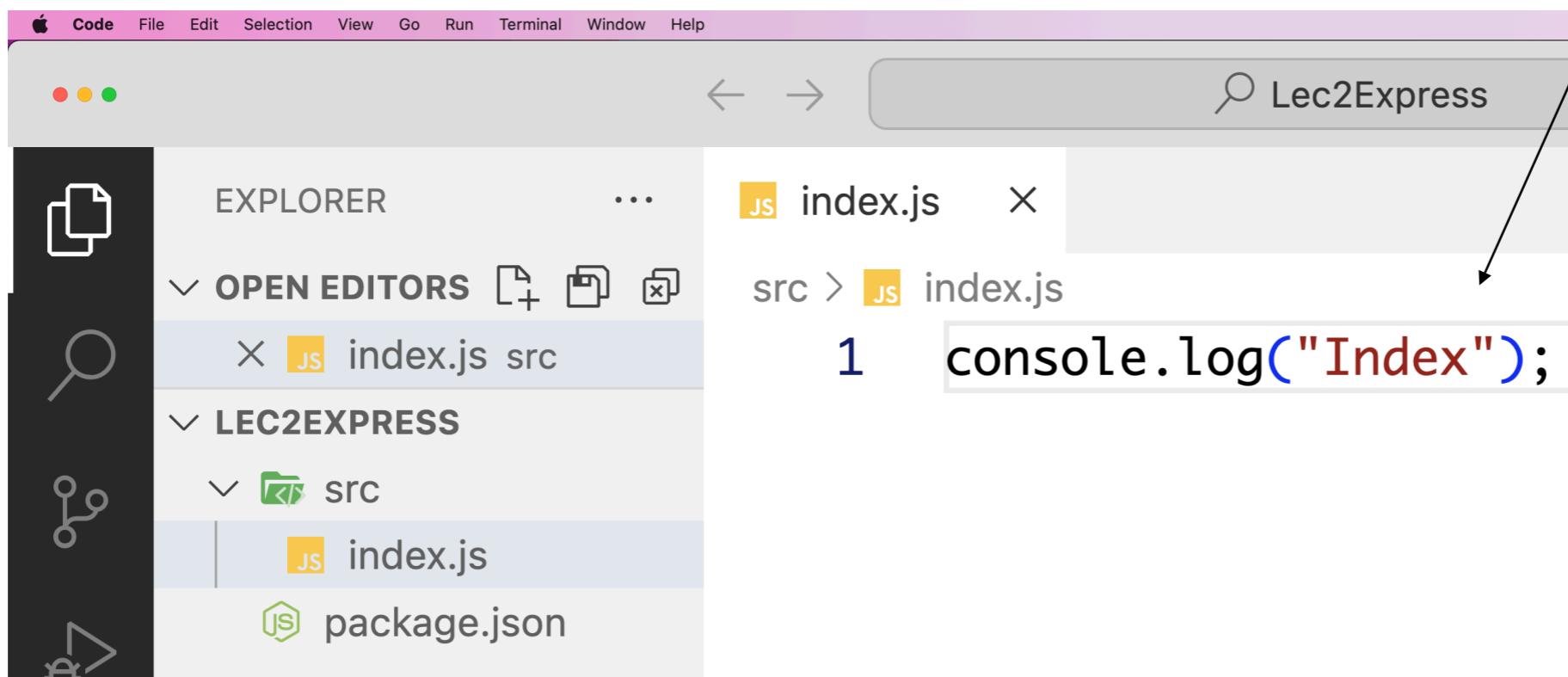
A screenshot of the VS Code interface. At the top, there are tabs for TERMINAL, PROBLEMS, and OUTPUT. Below the tabs, the TERMINAL tab is active, indicated by a dropdown arrow icon. The terminal window shows a file tree on the left with a folder named "Lec2Express" expanded. In the terminal itself, the command "/Users/T/W160523MR/" is displayed in blue, indicating it's a path. To the right of the path, the command "npm init -y" is being typed. A black circle with the number "3" is positioned to the right of the terminal window.

הספריה :Express



ניצור תיקיה בשם src
שבתוכה נכניס את קבצי הקוד שלנו:

4



ניצור קובץ js בתיקיה src

5

דרישה - בכל שינוי להריץ מחדש את הקוד:

```
npm i -D nodemon
```

6

```
{  
  "name": "lec2express",  
  "version": "1.0.0",  
  "description": "",  
  "main": "src/index.js",  
  "scripts": {  
    "start": "node src/index.js",  
    "watch": "nodemon src/index.js"  
  },  
  "keywords": [],  
  "author": "",  
  "license": "ISC"  
}
```

7

```
npm run watch
```

8

הספריה :Express

♥ Navigator Prefabricating Marinates

Pro Teams Pricing Document

npm Search packages

Search Sign Up

express DT

4.18.2 • Public • Published a year ago

[Readme](#) [Code](#) Beta [31 Dependencies](#) [74,919 Dependents](#) [270 Versions](#)

Install

```
> npm i express
```

Repository

[github.com/expressjs/express](#)

Homepage

[expressjs.com/](#)

Weekly Downloads

29,665,069

Version License

4.18.2 MIT

Unpacked Size Total Files

214 kB 16

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

npm v4.18.2 install size 1.89 MB downloads 120M/month

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

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

app.listen(3000)
```

npm i express

הספריה :Express

```
const express = require('express');  
  
// ~ const server = http.createServer();  
const app = express();  
  
app.get('/', (req, res) => {  
    res.end("Welcome to Express");  
});  
  
app.listen(8080);
```

יבוא של הספריה:

יצירת שרת (בדומה ל() (createServer))

ראוטר שמניג בבקשת כתובות:
http://localhost:8080/

http://localhost:8080

בחירה של פורט והרצה של השרת

הספריה :Express

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

// ~ const server = http.createServer();
const app = express();

app.get('/', (req, res) => {
  res.end("/");
});

app.get('/about', (req, res) => {
  res.end("about");
});

app.get('/home', (req, res) => {
  res.end("home");
});

app.listen(5500);
```

עם המודול של HTTP

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

const server = http.createServer();

const router = (req, res) => {
  if (req.url === '/home' && req.method === 'GET') {
    return res.end("Home");
  }
  if (req.url === '/about' && req.method === 'GET') {
    return res.end("about");
  }
  if (req.url === '/movies' && req.method === 'GET') {
    return res.end("movies");
  }
  res.end("Not Found");
}

server.on('request', router);

server.listen(5500);
```

הספריה :Express

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

const app = express();

app.get('/cards', (req, res) => {
  res.end("cards");
});

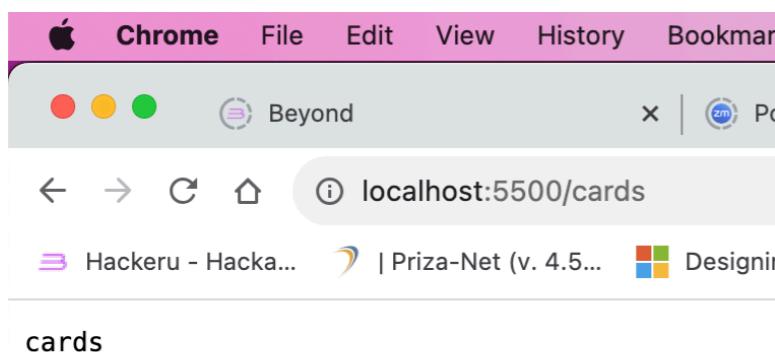
app.get('/users', (req, res) => {
  res.end("users");
});

app.listen(5500);
```

מסקנה:
Express

אין צורך במשפטים תנאי:

במקום זה: פונקציה בודדת מטפלת בEndpoint



הספריה :Express

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

const app = express();

const users = [
  {
    firstName: "John",
    lastName: "doe"
  },
  {
    firstName: "Jane",
    lastName: "doe"
  }
]

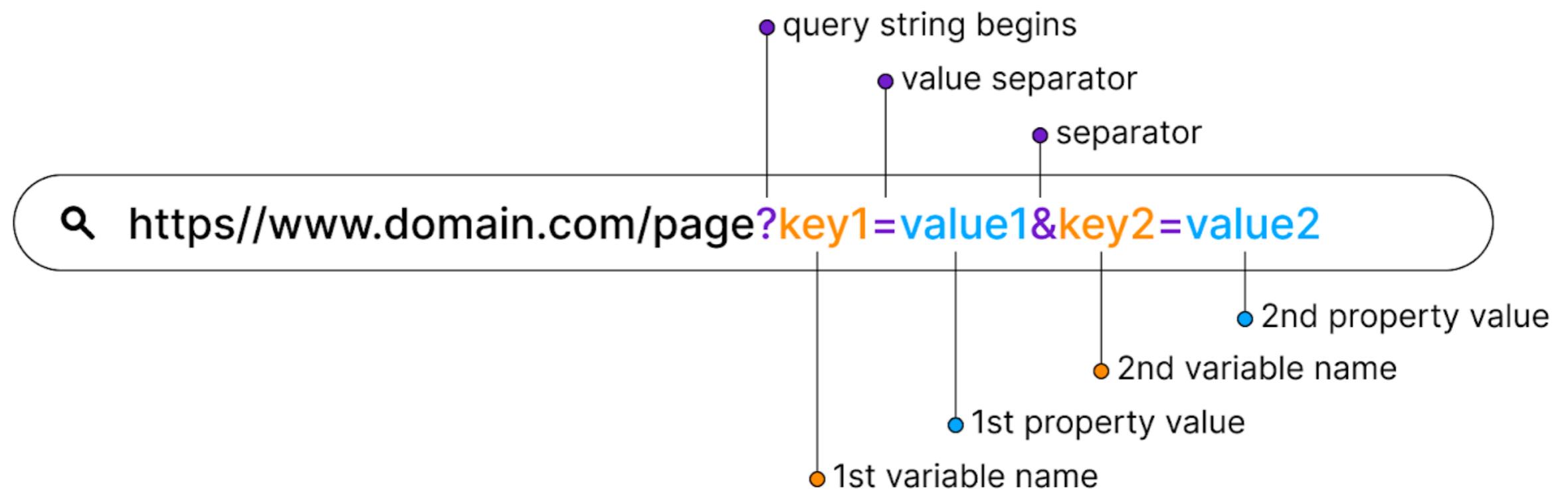
app.get('/users', (req, res) => {
  // res.end(JSON.stringify(users));
  res.json(users);
});

app.listen(5500);
```

מערך של אובייקטים:

<http://localhost:5500/users>

תזכורת לגביה Query String

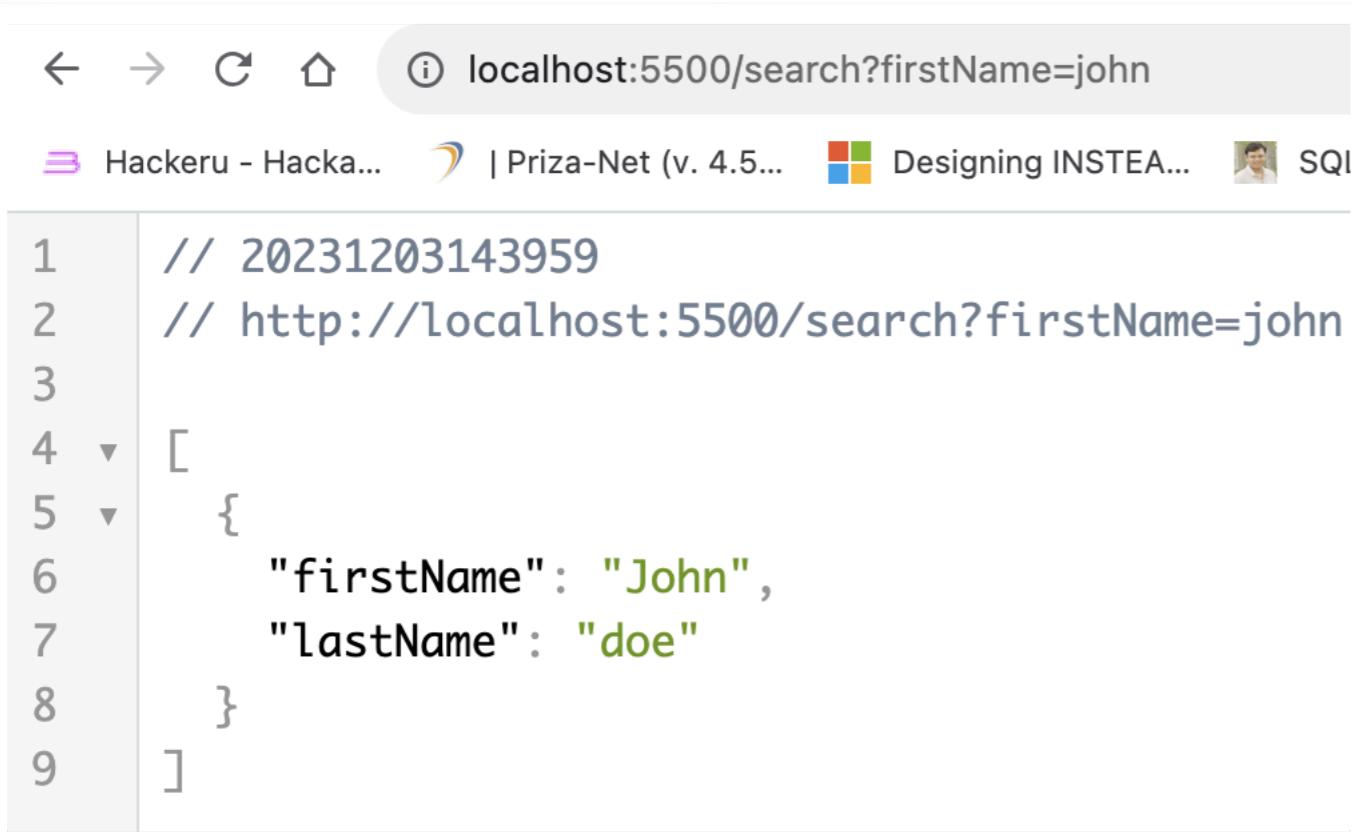


חיפוש:

```
//      /search?firstName=john
app.get("/search", (req, res) => {
  const { firstName } = req.query;
  return res.json(
    users.filter(
      (user) => user.firstName.toLocaleLowerCase() === firstName.toLocaleLowerCase()
    )
  )
})
```

Express
מטיפול אוטומטי ב Query String

Write less - do more - Express



A screenshot of a web browser window. The address bar shows the URL `localhost:5500/search?firstName=john`. Below the address bar, there are several tabs open: "Hackeru - Hacka...", "Priza-Net (v. 4.5...)", "Designing INSTEA...", and "SQL". The main content area displays a JSON array with one element:

```
1 // 20231203143959
2 // http://localhost:5500/search?firstName=john
3
4 [
5   {
6     "firstName": "John",
7     "lastName": "doe"
8   }
9 ]
```

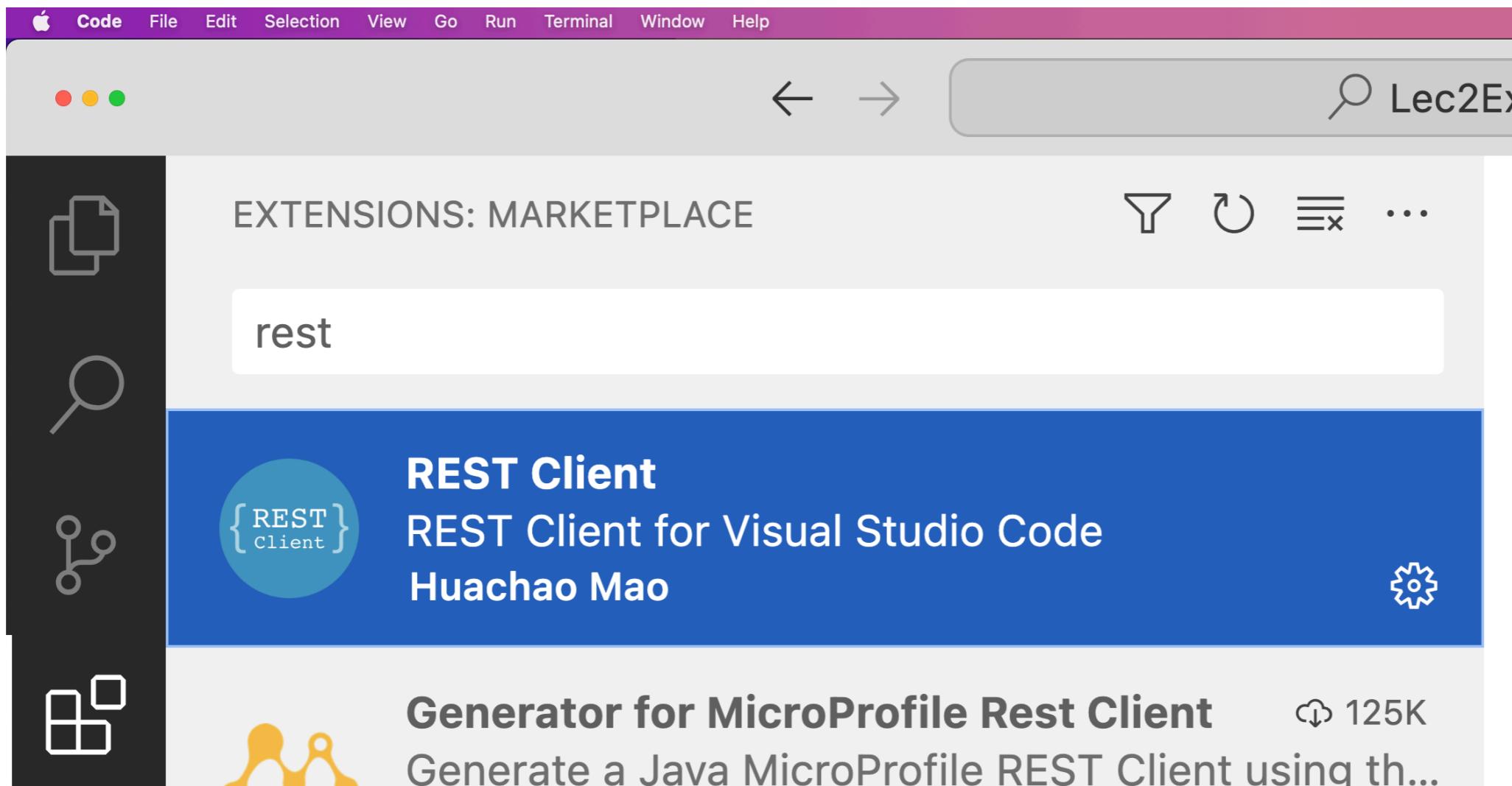
תזכורת לגביה Postman

The screenshot shows the Postman application interface. The left sidebar displays collections, environments, and history. The main workspace shows a collection named "Lec2" with a "users" folder. Inside the "users" folder, there is a "GET GET all Users" request. The request details show it's a GET method pointing to "http://localhost:5500/users". The "Headers" tab indicates 7 headers. The "Body" tab shows a JSON response with two users:

```
1 [ { "firstName": "John", "lastName": "doe" }, { "firstName": "Jane", "lastName": "doe" } ]
```

A status bar at the bottom right indicates "You are screen sharing" and includes a "Stop Share" button.

התקנה של תוסף:



עבודה עם התוסף:

ניצור קובץ עם סיומת .rest.

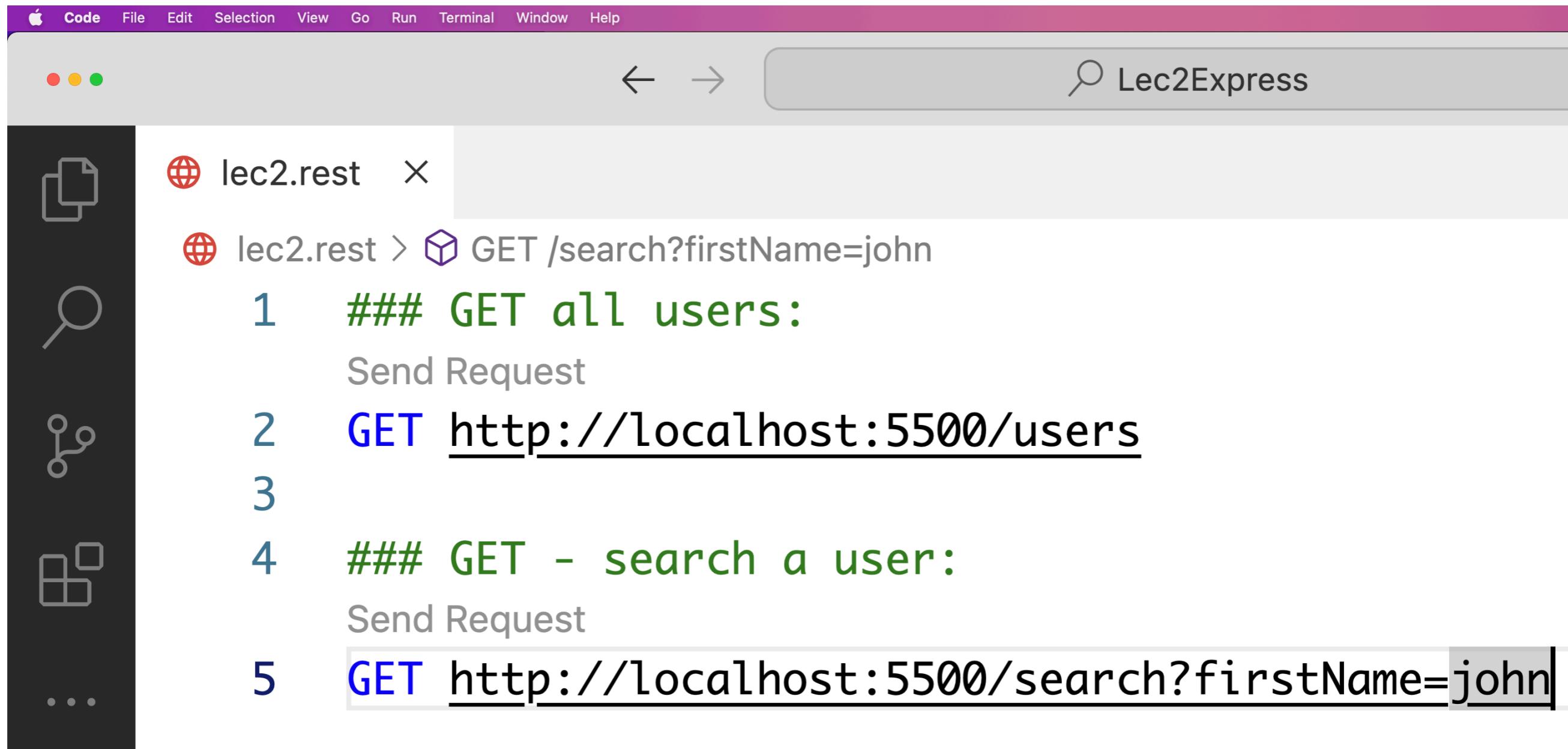
lec2.rest

The screenshot shows the Visual Studio Code (VS Code) interface. The top bar includes the Apple logo, Code, File, Edit, Selection, View, Go, Run, Terminal, Window, Help, and zoom buttons. The title bar says "Lec2Express". The left sidebar has icons for file, search, and code, with sections for EXPLORER, OPEN EDITORS, and LEC2EXPRESS. In the OPEN EDITORS section, there is a file named "lec2.rest". In the LEC2EXPRESS section, there are entries for "node_modules", "src", and another "lec2.rest" entry which is currently selected. The main editor area displays a REST API definition for "lec2.rest". It shows a GET request for "/users":

```
1  ### GET all users: ←  
2  GET http://localhost:5500/users
```

To the right of the editor, there is a status bar with the text "חוּבָה לְפָנֵי כָל בְּקַשָּׁה לְרִשּׁוֹם ##### וְלוּהוֹסִיף תִּיאוֹר".

שליחה של בקשות HTTP



A screenshot of a Mac OS X terminal window titled "Code". The window shows a session titled "lec2.rest" with the following content:

- 1 **### GET all users:**
Send Request
- 2 **GET http://localhost:5500/users**
- 3
- 4 **### GET - search a user:**
Send Request
- 5 **GET http://localhost:5500/search?firstName=john**

The terminal interface includes a sidebar with icons for file, search, and other functions.

בקשות POST

בקשת POST נרצה לשלוח body בformat JSON

lec2.rest ×

lec2.rest > POST /users

7 **### POST - add a user:**

Send Request

8 **POST http://localhost:5500/users**

לפחות שולח מידע בJSON
שולח Header שמצויר על כך.

9 **Content-Type: application/json**

10 ← שורה רוחה - סינטקס - סיום של Headers תחילת body

11 {

12 "firstName": "Bruce",
13 "lastName": "Wayne"

JSON חייב להיות תקני
مفתחות עם מרכאות
אין פסיק באיבר האחרון.

14 }

בקשות POST

בקשת POST נרצה לשלוח body בformat JSON

```
### POST - add a user:  
POST http://localhost:5500/users  
Content-Type: application/json
```

```
{  
  "firstName": "Bruce",  
  "lastName": "Wayne"  
}
```

התמודדות עם בקשות Post

ברירת מחדל - לא ידוע מה נרצה לקבל בBody
חייבים לציין מה הטיפוס שאמור להתקבל בbody:

הגדירה בExpress

SHIPUNA JSON שmagiu BODY

```
const app = express();

// configure express to handle incoming JSON body
app.use(express.json());
```

```
app.post('/users', (req, res) => {
  const user = req.body; ←
  users.push(user);

  res.json({ 'message': `User saved` , 'user': user})
})
```

בבקשת POST
/users

העדיין חיב להכיל body

{firstName:"", lastName: ""}

POST http://localhost:5500/users
Content-Type: application/json

```
{
  "firstName": "Bruce",
  "lastName": "Wayne"
}
```

בדיקה של http

אם נוריד את header content-type של req.body יהיה ריק.
express לא תמלא את הreq.body והוא יהיה ריק.

שיעור בית- Express

צרו אובייקט של שרת יאזוין בפורט 5500

השרת יאזוין לבקשת כתובות הבאות:

localhost:5500/about
localhost:5500/home
localhost:5500/frontend
localhost:5500/backend
localhost:5500/fullstack

בכל בקשה לאחד הדפים הללו בדף יש להחזיר מחרוזת מתאימה.

אם הלקוח מנסה לנוט לדף שלא הוגדר יש להחזיר בתגובה
404 - not found

רמז - app.get("*")
טופס את כל הבקשות שלא לטפלו בכלל כתובות

שיעור בית-**Express**

בקובץ `index.js` יש להגדיר מערך של `:todos`

```
const todos = [
  { task: "Complete Homework assignment", isComplete: false },
  { task: "Buy groceries", isComplete: false },
  { task: "Go for a run", isComplete: true }
];
```

`/todos`

GET

מחזיר את כל האובייקטים במערך.

`/todos`

POST מקבל אובייקט JSON בגוף הבקשת
ומוסיף אובייקט למערך

(בדומה למה שעשינו עם `users` בשקפים הקודמים במצגת)