

NodeJS

**הברות עם Npm ו Node
התקנה של מודולים ממקומם
מודולים מובנים
יצירת מודולים**

הקדמה: NodeJS

כדי לעבוד עם Node צריך להתקין אותו:

The screenshot shows the official Node.js website at nodejs.org/en/. At the top, there's a navigation bar with links like ABOUT, LEARN, DOWNLOAD, DOCS, GET INVOLVED, CERTIFICATION, and NEWS. Below the navigation, a banner states "Node.js® is an open-source, cross-platform JavaScript runtime environment." A "CYBER MONDAY" promotion is visible, offering up to 65% off. The main focus is the "Download Node.js®" section, which features two prominent buttons: "20.10.0 LTS Recommended For Most Users" and "21.2.0 Current Latest Features". Arrows from the Hebrew text on the left point to these buttons.

גירסה יציבה → **20.10.0 LTS**
Recommended For Most Users

← גירסה נסionaית **21.2.0 Current**
Latest Features

For information about supported releases, see the [release schedule](#).

הקדמה: NodeJS

Node.js® is an open-source, cross-platform **JavaScript runtime environment.**

מתאים לכל מע' הפעלה

מאפשר לכתב קוד בJS
ולהרץ אותו שירותים מול Node
(בלי דפדפן)

לפני NodeJS

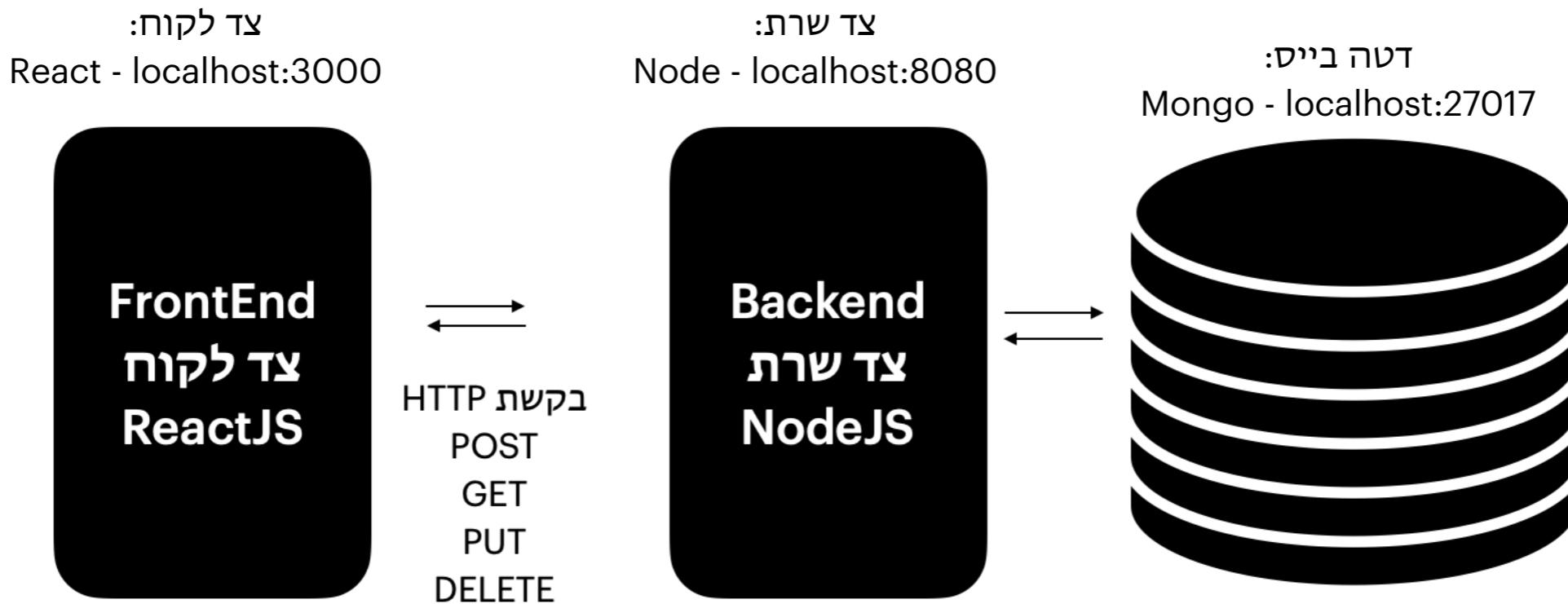
JS היה משמש רק הצד לקוח בדף
הדף היה מרים קוד JS

כיום אפשר לכתב כל תוכנה
בNodeJS - דוגמא - VSCode כתובה בJS

הקדמה:

NodeJS

במודול זהה - נשתמש בNode לכתיבת צד שרת

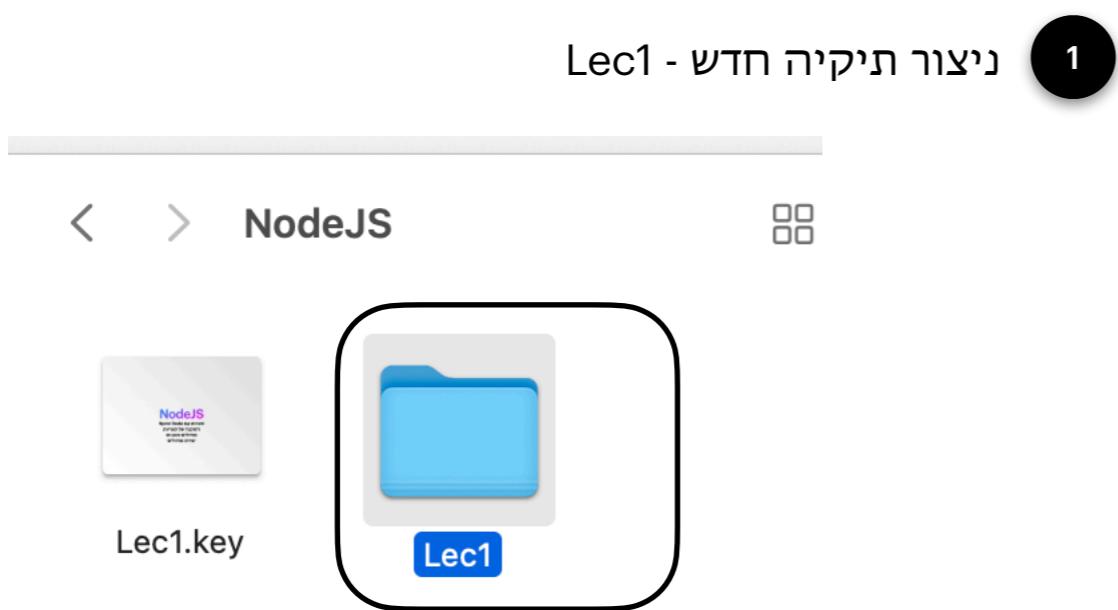


בזמן פיתוח - מחשב אחד שMRIIZ את כל השירותים (צד-לקוח/צד-שרת וDatabasefricanus)

במציאות - 3 מחשבים שונים (לפחות)

בפרויקטים גדולים יותר - יכולות להיות מספר מכונות שMRIIZות כל חלק

פרויקט חדש: NodeJS



2 נגרור את התקינה לVSCode
ונפתחה הטרמינל בVSCode

3 בטרמינל:
npm init -y
הפקודה יוצרת את הקובץ package.json

npm init
שואלהות אונთן שאלות
npm init -y
עונה על כל השאלות

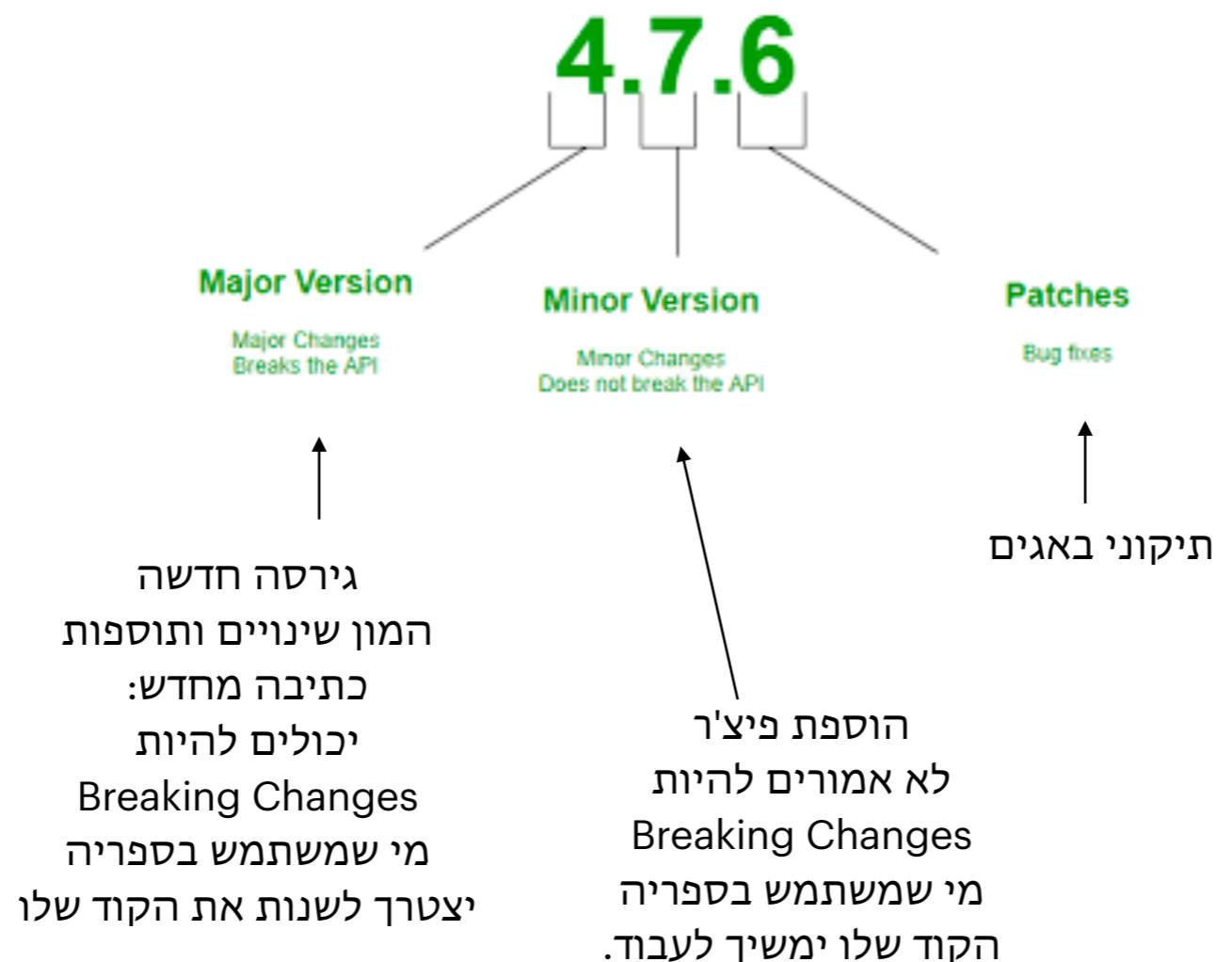
package.json

קובץ שמתאר את הפרויקט:

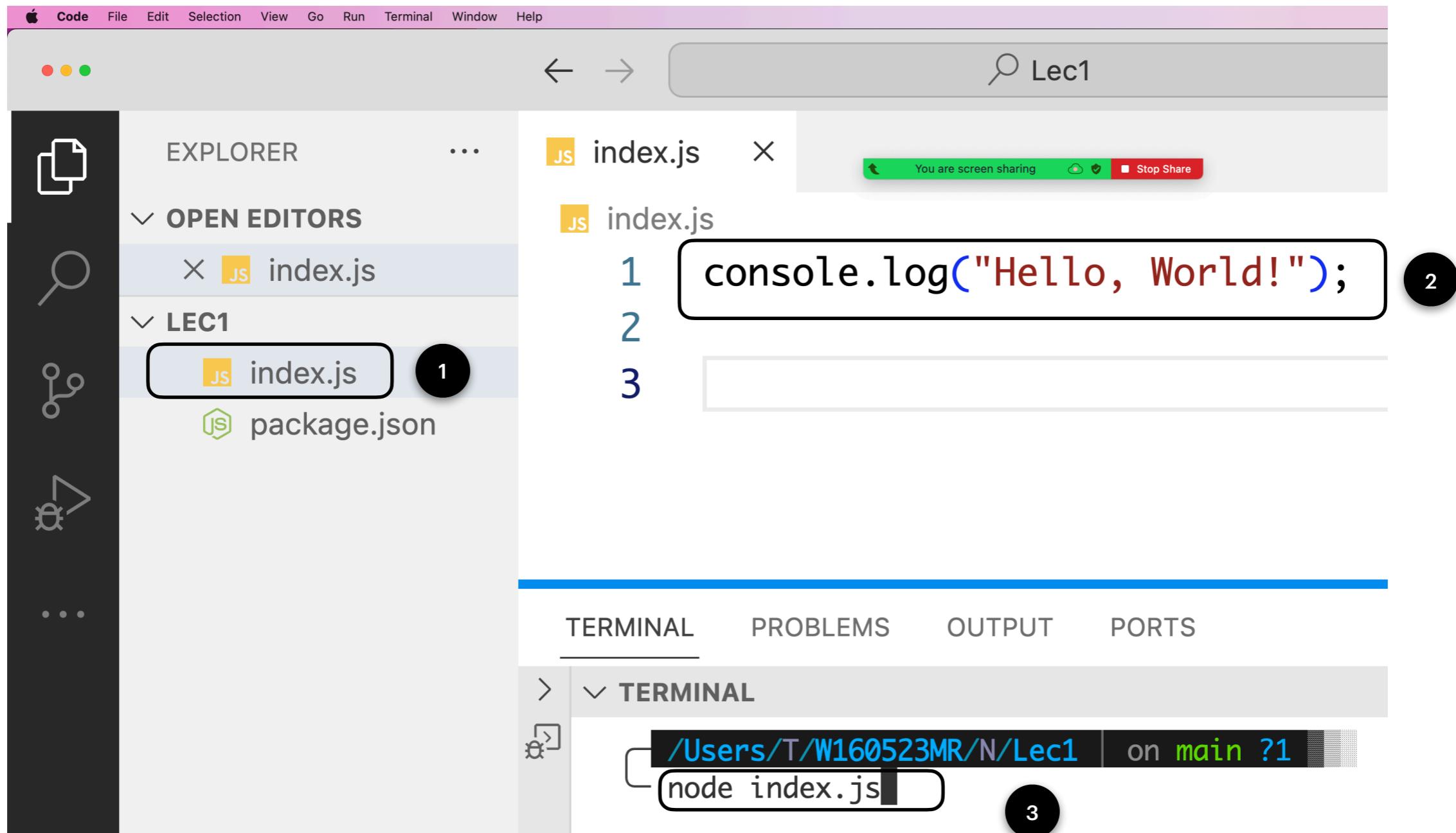
```
{  
  "name": "lec1",  
  "version": "1.0.1",  
  "description": "intro to node js",  
  "main": "index.js", ← נקודת כניסה - קובץ ראשי  
  "scripts": {  
    "test": "echo \\\"Error: no test specified\\\" && exit 1" ← קוד שMRIIZ את התוכנית  
  },  
  "keywords": ["lec1", "intro"],  
  "author": "Hackeru",  
  "license": "ISC"  
}
```

אפשר להעלות את הפרויקט כחבילת וpkg
התיאור בקובץ ישמש את pkg כדי להציג את החבילה

package.json



הרצה של קובץ



Hello, World!

**סקריפט להרצה של הפרויקט שלנו:
משמש גם למטרות תיעוד של הפרויקט:**

```
{  
  "name": "lec1",  
  "version": "1.0.0",  
  "description": "",  
  "main": "index.js",  
  "scripts": {  
    "start": "node index.js"  
  },  
  "keywords": [],  
  "author": "",  
  "license": "ISC"  
}
```

npm run start
npm start

Node מנהל חבילות Npm

מאפשר להתקין ולהריץ חבילות

The screenshot shows a Chrome browser window with the address bar displaying 'npmjs.com/package/uuid'. The page content is the npm package page for 'uuid'. At the top, there's a navigation bar with links like 'Readme', 'Code', 'Dependencies', 'Dependents', 'Versions', 'CI', 'Browser', and 'Fund this package'. Below the navigation, there's a note about upgrading from version 3. The page also includes sections for 'Install', 'Repository', 'Homepage', 'Weekly Downloads', 'Version', and 'License'.

uuid DT

9.0.1 • Public • Published 3 months ago

[Readme](#) [Code](#) Beta [Dependencies](#) [Dependents](#) [37 Versions](#)

uuid CI passing Browser passing

For the creation of [RFC4122](#) UUIDs

- **Complete** - Support for RFC4122 version 1, 3, 4, and 5 UUIDs
- **Cross-platform** - Support for ...
 - CommonJS, [ECMAScript Modules](#) and [CDN builds](#)
 - NodeJS 12+ ([LTS releases](#))
 - Chrome, Safari, Firefox, Edge browsers
 - Webpack and rollup.js module bundlers
 - [React Native / Expo](#)
- **Secure** - Cryptographically-strong random values
- **Small** - Zero-dependency, small footprint, plays nice with "tree shaking" packagers
- **CLI** - Includes the `uuid` command line utility

Note Upgrading from `uuid@3`? Your code is probably okay, but check out [Upgrading From `uuid@3`](#) for details.

Install

```
> npm i uuid
```

Repository

Homepage

Weekly Downloads

96,200,301

Version

9.0.1

License

MIT

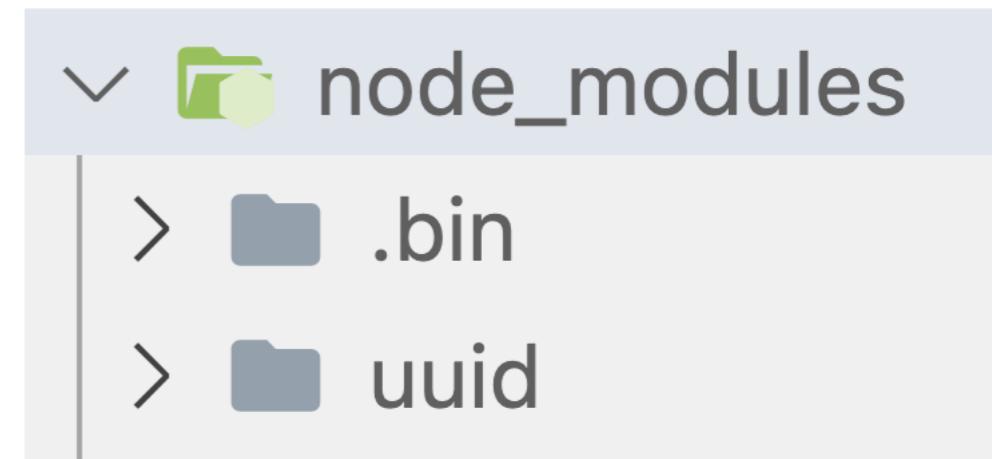
Npm ניהול חבילות ל-node

```
{  
  "name": "lec1",  
  "version": "1.0.0",  
  "description": "",  
  "main": "index.js",  
  "scripts": {  
    "start": "node index.js"  
  },  
  "keywords": [],  
  "author": "",  
  "license": "ISC",  
  "dependencies": {  
    "uuid": "^9.0.1"  
  }  
}
```

הורדה של חבילה:

```
npm i uuid
```

החבילה ירדה למחשב שלנו:
(נמצאת בתיקיה node_modules)

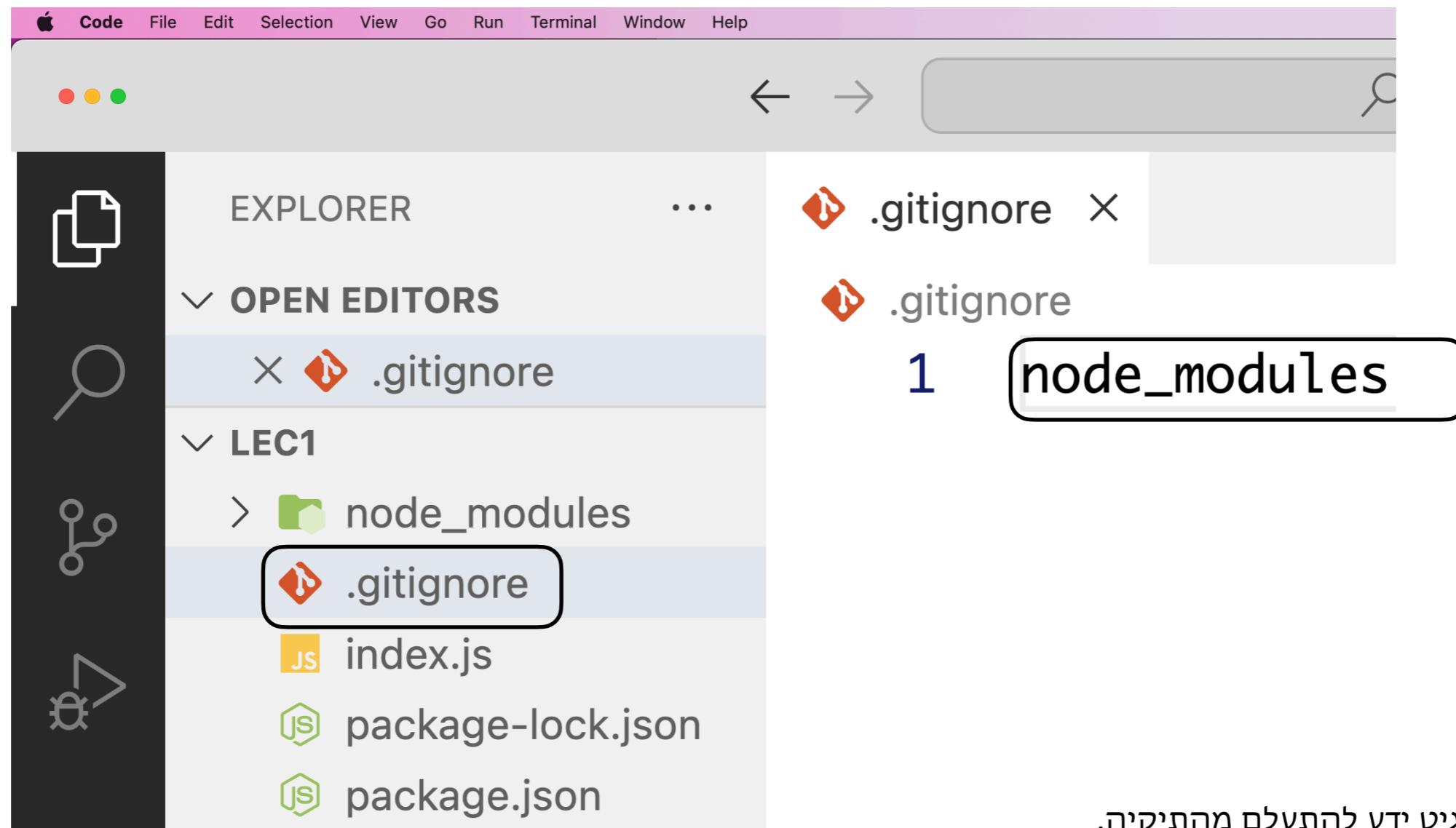


החבילה נספה לקובץ package.json

אם נמחק את התיקייה
node_modules
npm install עם אותה שם
אפשר לשחזר אותה

לא נעלת את התקינה `node_modules` לגיט/גיטהאב

1) ניצור קובץ חדש בשם `.gitignore`



גיט ידע להעתים מהתקינה.

ייבוא של מודול חיצוני ב-node

התקנו את המודול `uuid`

שימוש ב-Node Modules

```
const uuid = require('uuid');  
const result = uuid.v4();  
console.log(result);
```

ייבוא של מודול

object uuid עם מתודות
נפעיל את הפונקציה `v4`

npm start

> node index.js

אותו דבר עם

564a8ae3-1621-453c-82b0-094d0632a7bd

```
import uuid from 'uuid';  
  
const result = uuid.v4();  
  
console.log(result);
```

עוד דוגמא להתקנת מודול:

npm i qrcode

```
const Qrcode = require('qrcode');

Qrcode.toDataURL(text = "Welcome to Node", (err, url) => {

  if (err) {
    console.error(err)
  } else {
    console.log(url);
  }
})
```

npm start

עד דוגמא להתקנת מודול:

npm i qrcode

```
const Qrcode = require('qrcode');

// print the dataurl to the console
Qrcode.toDataURL(text = "Welcome to Node", (err, url) => {
  if (err) {
    console.error(err)
  } else {
    console.log(url);      נסתכל בדוקומנטציה של הספרייה במקו
                            ונמצא איך משתמשים בספריה:
  }
})

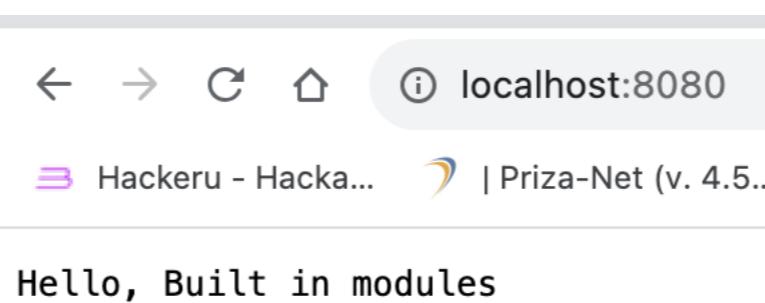
// print the qrcode to the terminal:
Qrcode.toString('I am a pony!', { type: 'terminal' }, function (err, url) {
  console.log(url)
})

// print the qrcode to a file:
Qrcode.toFile("pony.png", "Im a Pony!", (err) => {
  if (err) {
    console.error(err);
  }
})
```

מודולים מובנים ב-node.js

אין צורך בהורדה מnpm - מובנה ב-node.js עצמה.

מודול מובנה שיעזר לייצור שרת http



```
// http is a built in module - don't need to npm it
const http = require('http');

const server = http.createServer((req, res) => {
  res.end("Hello, Built in modules");
});

//start the server on port 8080:
server.listen(8080);
```

מודולים מובנים ב-node.

אין צורך בהורדה מnpm - מובנה ב-node עצמה.

//sync: runs line by line (regular code)

//async: callback, promise

```
const readline = require('node:readline');
const { stdin: input, stdout: output } = require('node:process');

const rl = readline.createInterface({ input, output });

rl.question('What do you think of Node.js? ', (answer) => {
    // TODO: Log the answer in a database
    console.log(`Thank you for your valuable feedback: ${answer}`);
    rl.close();
});
```

עובדת עם Callback

הfonקציה question
מקבלת 2 פרמטרים:

```
question : string
callback: (ans:string)=>void
```

קוד אסינכריוני:
fonkציית callback
תופעל ברגע שתתקבל תשובה.

מודולים מובנים ב-node

גישה - הפונקציה תופעל אחרי שהפעולה האסינכרונית תסתיים

```
//import the readline built in module:  
const readline = require('node:readline');  
  
//import the standard input/output streams:  
//stdin = keyboard  
//stdout = console  
const { stdin, stdout } = require('node:process');  
  
//config readline to use stdin (keyboard), stdout (console)  
const rl = readline.createInterface({ input: stdin, output: stdout });  
  
//the question will be printed to the console (stdout)  
// the answer will be captured from the keyboard(stdin)  
rl.question('What do you think of Node.js? ', (answer) => {  
    // TODO: Log the answer in a database  
    console.log(`Thank you for your valuable feedback: ${answer}`);  
  
    rl.close();  
});
```

תזכורת לגבי עבודה עם Promises

```
const demoFetch = async () => {
  try {
    const res = await fetch('');
  }
  catch (e) {

  }
}

const demoFetch2 = () => {
  fetch('')
    .then(res => console.log(res))
    .catch(e => {
      console.log(e);
    })
}
```

מודולים מובנים ב-node: גישת Promise - משנה את המבנה של הקוד:

הmethod `Promise<string> question()` מוחזירה
כדי לקבל ממנה את התשובה - נשתמש ב`await` או ב`(then)`

```
const readline = require('node:readline/promises');
const { stdin, stdout } = require('node:process');
const rl = readline.createInterface({ input: stdin, output: stdout });

const demoReadline = async () => {
  const ans = await rl.question("What do you think of Node.js?")
  console.log(`Thank you for your valuable feedback: ${ans}`);
  rl.close();
}

demoReadline();
```

סיכום ביניים:

התקנה של מודולים
מודולים מובנים

אין צורך להכיר את המודולים `qrcode` או `pin`

את המודול `http` נלמד בלבד - רק ראיינו שהוא קיים.
המודול `readline` - שימושו כדי להציג 2 גישות callback/promise

רשימה חלקית של מודולים מבנים:

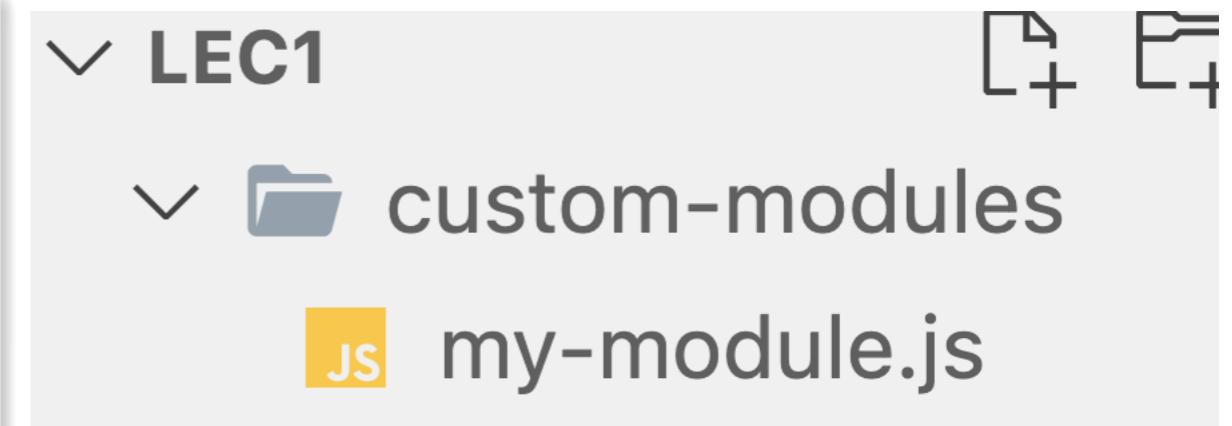
const os = require('os');	מידע על המעבד, מע' הפעלה, זכרו פנו:
const http = require('http');	עבודה עם בקשות http ליצירת צד-שרת:
const fs = require('fs');	עבודה עם קבצים: קריאה וכתיבה של קבצים:
const readline = require('readline');	קלט מהמשתמש:
const path = require('path');	נתיב לקובץ: לדוגמא נתיב לתיקית images

מודולים:

לא נכתבת תוכנה שלמה בקובץ אחד

יצירת מודול משלנו:

```
class Person {  
  //props:  
  name;  
  
  constructor(name) {  
    this.name = name;  
  }  
  
  sayHello() {  
    return `Hello, My name is: ${this.name}`  
  }  
}  
  
//~ export default Person  
module.exports = Person;
```



מודולים:

ייבוא של מודול:

```
const Person = require('./custom-modules/my-module');

const p1 = new Person('John');

console.log(p1.sayHello())
console.log(p1.name)
```

Hello, My name is: John
John

מודולים:

יצוא של מספר אובייקטים מודול:

```
class Person {  
  //props:  
  name;  
  
  constructor(name) {  
    this.name = name;  
  }  
  
  sayHello() {  
    return `Hello, My name is: ${this.name}`  
  }  
}  
  
class PersonList {  
  // properties:  
  people;  
  
  // constructor:  
  constructor() {  
    this.people = [];  
  }  
  
  //methods:  
  addPerson(person) {  
    this.people.push(person);  
  }  
  
  getPeople() {  
    return this.people;  
  }  
}  
  
const foo = 'bar';  
module.exports = { Person, PersonList, foo };
```

מחלקה:
תכונות, בנאי, פעולות

מודולים:

יבוא של מספר אובייקטים ממודול:

```
const { Person, PersonList } = require('./custom-modules/my-module');
```

```
const personList = new PersonList();
```

```
personList.addPerson(new Person('John'));
personList.addPerson(new Person("Bob"));
personList.addPerson(new Person("Jane"));
```

```
personList.getPeople().forEach(person => {
  console.log(person.sayHello());
  console.log(person.name);
});
```

מודולים:

דוגמא למתודות סטטיות:

Date.now()

Math.random()

מתודה סטטית: מתודה שלא עושה שימוש בתכונות של האובייקט
ולבן לא יוצרים מופע - מפעילים ישירות את המתודה.

Utils.random(20)

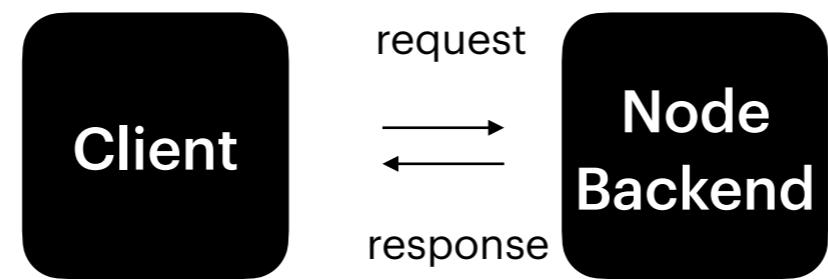
```
class Utils {  
    static random(max = 10) {  
        return Math.round(Math.random() * max)  
    }  
}  
  
module.exports = Utils;
```

מתודה מופע: מתודה שיכולה להשתמש בתכונות של האובייקט -
לדוגמא המתודה sayHello - מציגה את name של האובייקט.

```
const person = new Person('John');  
person.sayHello();
```

מודול מובנה - המטרה - ליצור שרת HTTP

(במציאות נשתמש בספריה בשם Express
ספריה שמאוד מקלה על העבודה עם HTTP)

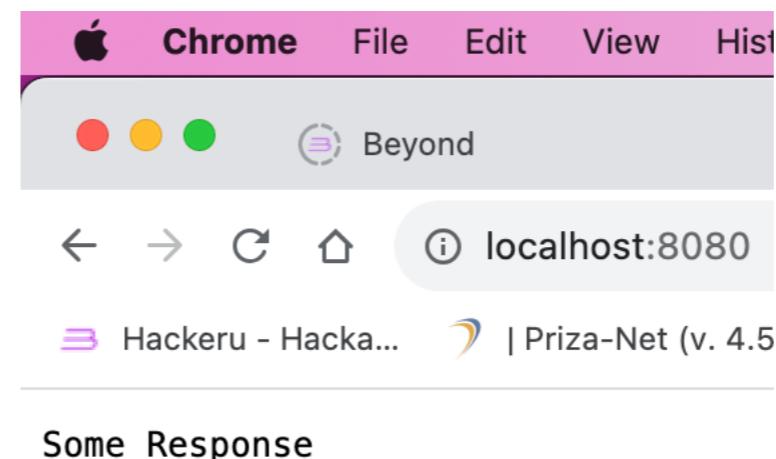


מדריך מבנה - Http

```
// ייבוא של המודול המובנה
const http = require('http');

// שרת - לקבל בקשה - ולהזידר תגובה
const server = http.createServer((req, res) => {
  res.end("Some Response");
});

//start the server on port 8080
server.listen(8080);
```



מודול מובנה - Http

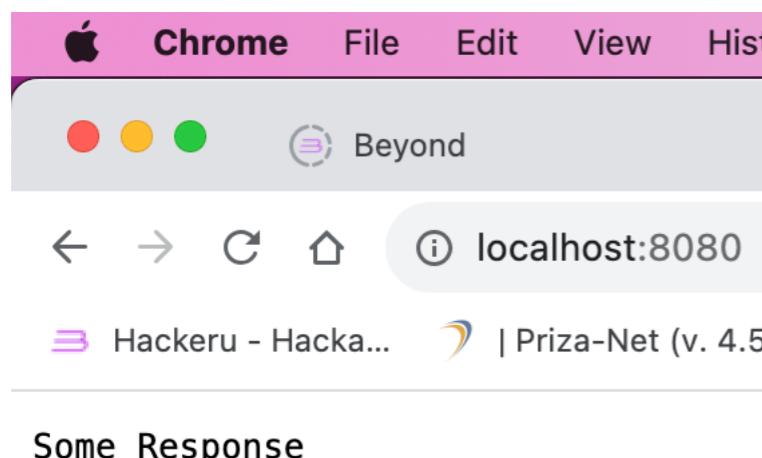
```
// יבוא של המודול המובנה
const http = require('http');

// שרת - לקבל בקשה - ולהזיר תגובה
const server = http.createServer((req, res) => {
  res.end("Same Response");
});

//start the server on port 8080
server.listen(8080);
```

בעה -
כשמשנים קוד - צריך לעצור את התכנית:
ctrl+c

ולהפעיל שוב npm start



3 דרכים להתקיןnodemon:

מתווסף לחבילות בpackage.json

npm i express

```
"dependencies": {  
  "nodemon": "^3.0.1",  
  "qrcode": "^1.5.3",  
  "uuid": "^9.0.1"  
}
```

npm i -g nodemon

מתקין חבילה באופן גלובלי על המחשב - ללא קשר לפרוייקט הנוכחי.

מה מותקן גלובלית:

החבילה לא תתווסף לpackage.json

npm list -g

nodemon מותקנת גלובלית על המחשב.

אפשר לרשום בכל טרמינל במחשב nodemon והפקודה תרוץ.

npm i -D nodemon

רושם את החבילה בpackage.json

Dev Dependency

```
"devDependencies": {  
  "nodemon": "^3.0.1"  
}
```

(אפשר לנו להפריד חבילות שדרושים רק לפיתוח)

במוצר הסופי - לא צריך את ההתקנה הזאת.

עבודה עם NodeMon

npm i -D nodemon

package.json

```
"devDependencies": {  
  "nodemon": "^3.0.1"  
}
```

package.json

```
"scripts": {  
  "start": "node index.js",  
  "watch": "nodemon index.js"  
},
```

נוסיף סקריפט

אפשר לקרוא לסקריפט dev

npm run watch

npm run start

npm start

החזרת תגובה בהתאם לrequest

```
// ייבוא של המודול המובנה
const http = require('http');

// שרת - מקבל בקשה - ולהציגו תגובה
const server = http.createServer((req, res) => {
    if (req.url === '/home') {
        return res.end("Home");
    }
    if (req.url === '/about') {
        return res.end("about");
    }
    if (req.url === '/movies') {
        return res.end("movies");
    }

    res.end("Not Found");
});

//start the server on port 8080
server.listen(8080);
```

נוסיף סקריפט

דרך נוספת להגיב לבקשת

```
// ייבוא של המודול המובנה
const http = require('http');

// שרת - מקבל בקשה - ולהציג תגובה
const server = http.createServer((req, res) => {
    if (req.url === '/home') {
        return res.end("Home");
    }
    if (req.url === '/about') {
        return res.end("about");
    }
    if (req.url === '/movies') {
        return res.end("movies");
    }

    res.end("Not Found");
});

//start the server on port 8080
server.listen(8080);
```

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

const server = http.createServer();

server.on('request', (req, res) => {
    if (req.url === '/home') {
        return res.end("Home");
    }
    if (req.url === '/about') {
        return res.end("about");
    }
    if (req.url === '/movies') {
        return res.end("movies");
    }

    res.end("Not Found");
})

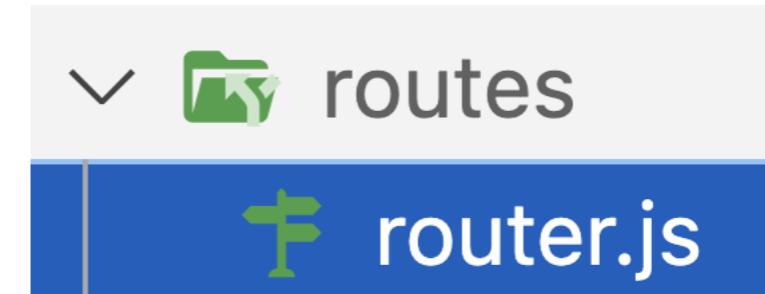
server.listen(8080);
```

מודולויזה: ניצור קובץ חדש עבור פונקציה

routes/router.js

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

module.exports = router;
```



index.js

```
const router = require('./routes/router');
const http = require('http');

const server = http.createServer();

server.on('request', router);

server.listen(8080);
```

עד מודולים

לכל אירוע יכולם להיות מספר מאזינים:

▽  middleware
 └  logger.js

```
const logger = (req, res) => {
  console.log(req.url);
  console.log(req.method);
}

module.exports = logger;
```

SRP = Single Responsibility Principle:
עקרון האחוריות הבודדת:

לכל מודול ולכל פונקציה ולכל מחלקה תפקיד אחד.

index.js

```
const logger = require('./middleware/logger');
const router = require('./routes/router');
const http = require('http');

const server = http.createServer();

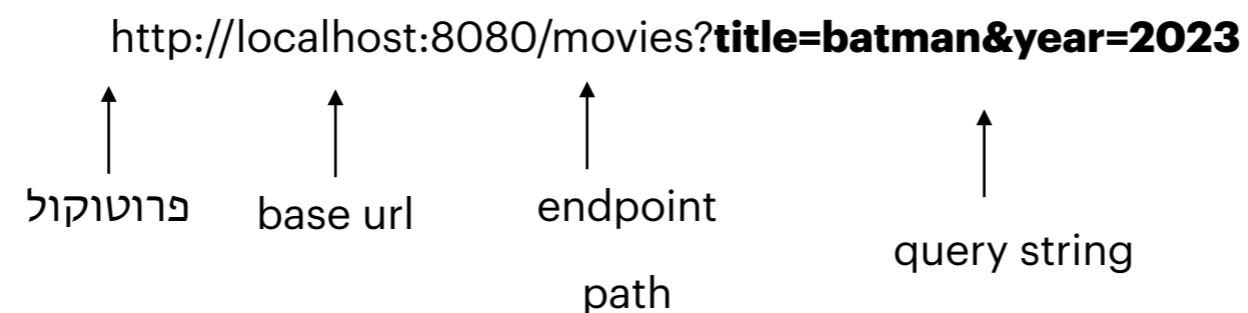
server.on('request', logger);
server.on('request', router);

server.listen(8080);
```

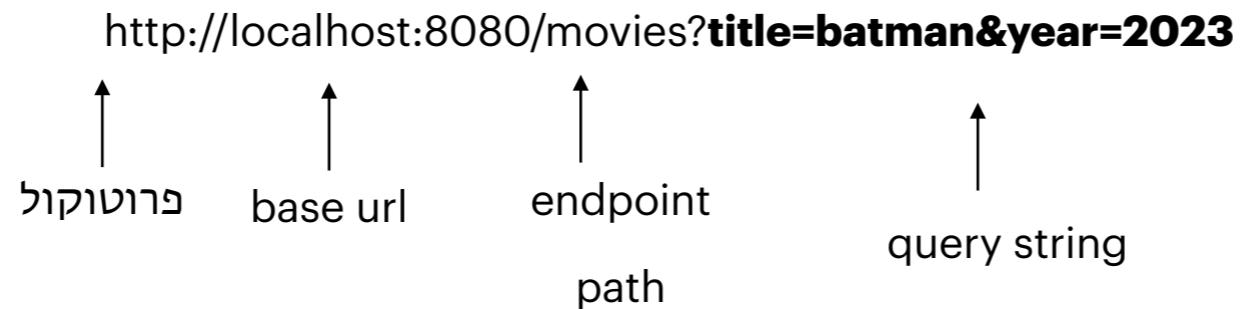
Middleware



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 תוקן שימוש בספרייה underscore
יש לקרוא בדוקומנטציה של הספרייה לגבי הפעולה `uniq` ולהשתמש בה.

משימת
חיפוש

שיעור בית - HTTP:

צרו אובייקט של שרת שמאזין בפורט 5000

השרת יאוזן לבקשת 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

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