Node.Js

Routing

01/12/2024

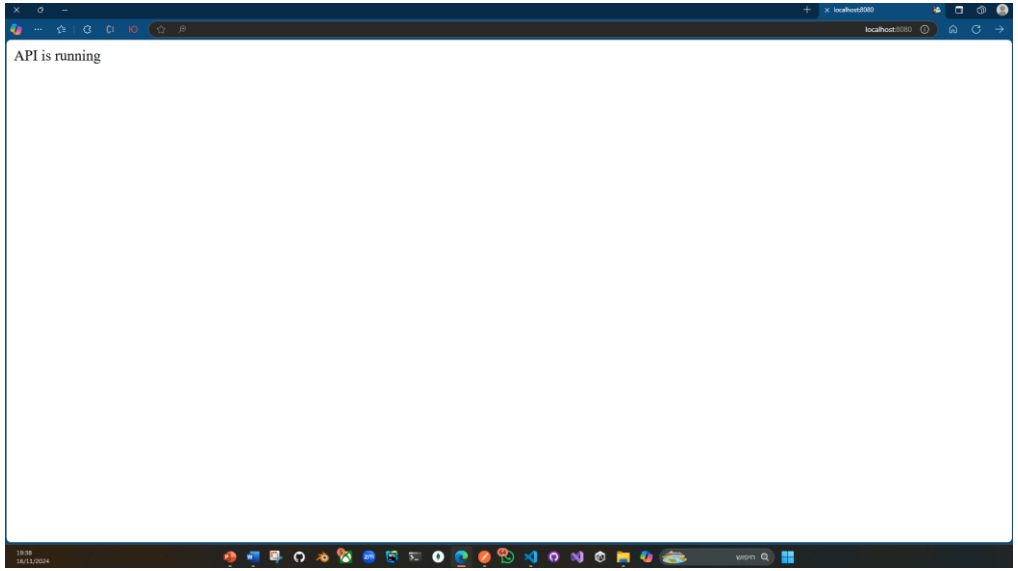
REST API's

REST API's

בחלק זה נכיר ונלמד מה הוא **REST API** וניצור את ה-**API** הראשון שלנו באמצעות שימוש ב-**Express**.
בסיום הנושא תוכלו לענות על השאלות הבאות:

- מה הם שרתי **REST (Representational State Transfer API)**?
- מה הם פקודות ה-**HTTP** הנפוצות בשרתי **REST**?
- כיצד נוכל להגדיר בתוכנית שלנו נתיב שיגיב לבקשות **GET**?
- כיצד נוכל להגדיר בתוכנית שלנו נתיב שיגיב לבקשות **POST**?
- כיצד נוכל להגדיר בתוכנית שלנו נתיב שיגיב לבקשות **PUT**?
- כיצד נוכל להגדיר בתוכנית שלנו נתיב שיגיב לבקשות **DELETE**?
- מה עלינו לשלוח כפונקציית callback בעת שימוש במתודות אלו?

```
app.get('/', (req, res) => {
    res.send('API is running');
});
```



REST API's – HTTP Methods

בחלק זה נכיר את **nodemon** ונדון בהתקנתה. בסיום הנושא תוכלו לענות על השאלות הבאות:

- מה משמעות הפרמטרים **req** ו-**res**?
- לשם מה נשתמש במתודה "**send**"?
- מה עלינו לתת לה כפרמטר?
- מה אם ברצוננו להחזיר אובייקט או מערך?
- לשם מה נשתמש במתודה "**Json**"?
- לשם מה נשתמש במתודה "**status**" ומה עלינו לשלוח לה כפרמטר?

```
app.get('/hello', (req, res) => {
    res.json({
        date: new Date().toLocaleDateString(),
        message: 'Hello, World!'
    })
});
```

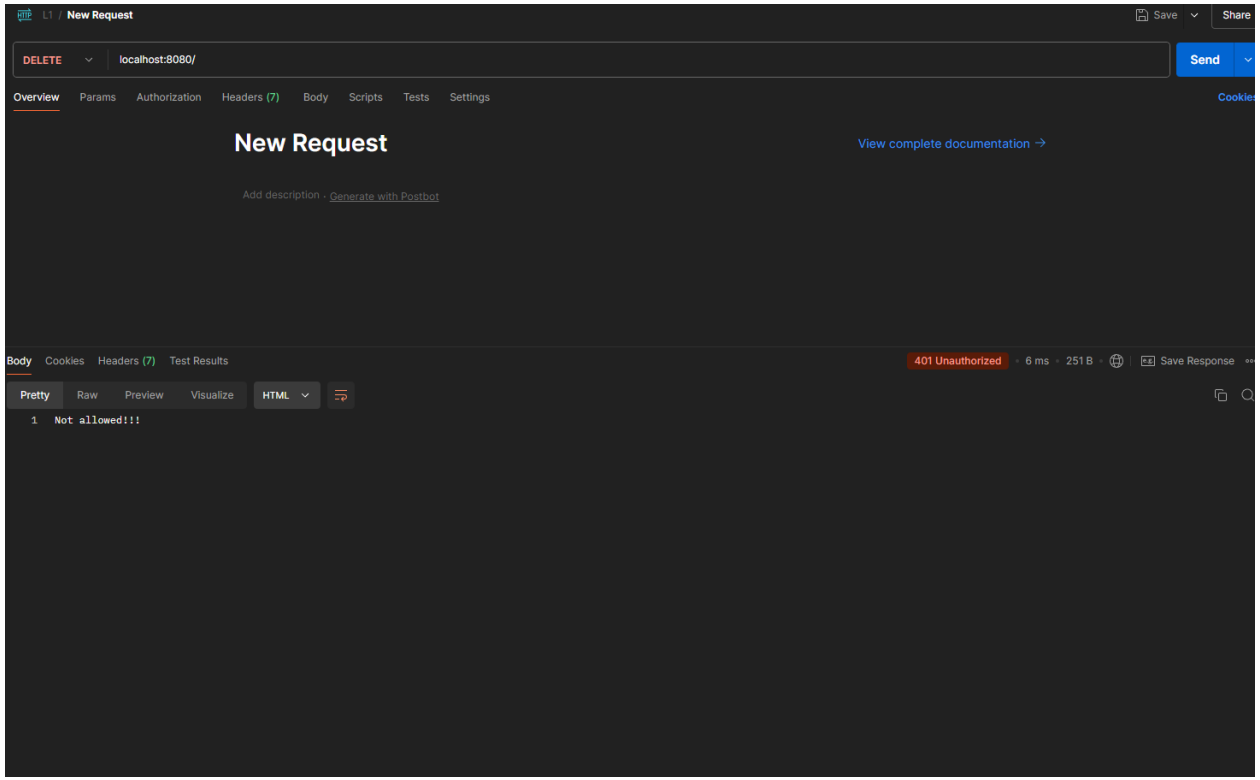
```
{
    "date": "16.11.2024",
    "message": "Hello, World!"
}
```

```
app.delete('/', (req, res) => {
    res.status(401).send('Not allowed!!!');
});
```

Welcome/biz_cards_dev.rar

REST API's - תרגול

צרו תוכנית **Node.js** חדשה עבור הנושא ופתרו את התרגילים לפי הסדר.



תרגיל	תיאור המשימה
Ex-1	צרו נתיב בשם "**hello**" המגיב לבקשות **GET**, כתשובה יחזיר אובייקט המכיל שדה עבור התאריך, ושדה עבור הודעה שתברך את המשתמש לשלום.
Ex-2	בידקו את תפקוד ה-**API** באמצעות שימוש בדפדפן.
Ex-3	צרו נתיב בשם "**ping**" המגיב לבקשות **POST**, כתשובה יחזיר את ההודעה "**pong**".
Ex-4	צרו נתיב ללא שם המגיב לבקשות **DELETE**, כתשובה יחזיר את ההודעה " **Not Allowed!!**" עם סטטוס **401**.
Ex-5	בידקו את תפקוד ה-**API** באמצעות שימוש באפליקציה **Postman**.

ניהול Routing ב-Express.js

ניהול Routing ב-Express.js

בחלק זה נלמד כיצד ניתן לנהל ניתוב **(Routing)** באפליקציית **Express**.
בסיום הנושא תוכלו לענות על השאלות הבאות:

- מדוע שנרצה לנהל את כל הניתוב במודול נפרד?
- כיצד נוכל ליצור נתב (**Router**) במודול נפרד?
- מדוע עלינו לייבא את **Router** לשם כך?
- מה עלינו לייצא מקובץ זה?
- כיצד ניתן להוסיף את הנתב (**Router**) לקובץ ההרצה הראשי של השרת?

```
import { Router } from "express";

const router = Router();

router.get("/", (req, res) => {
    res.send("Router is working");
});

export default router;
```

```
// Add the router to the app
app.use(router);
```

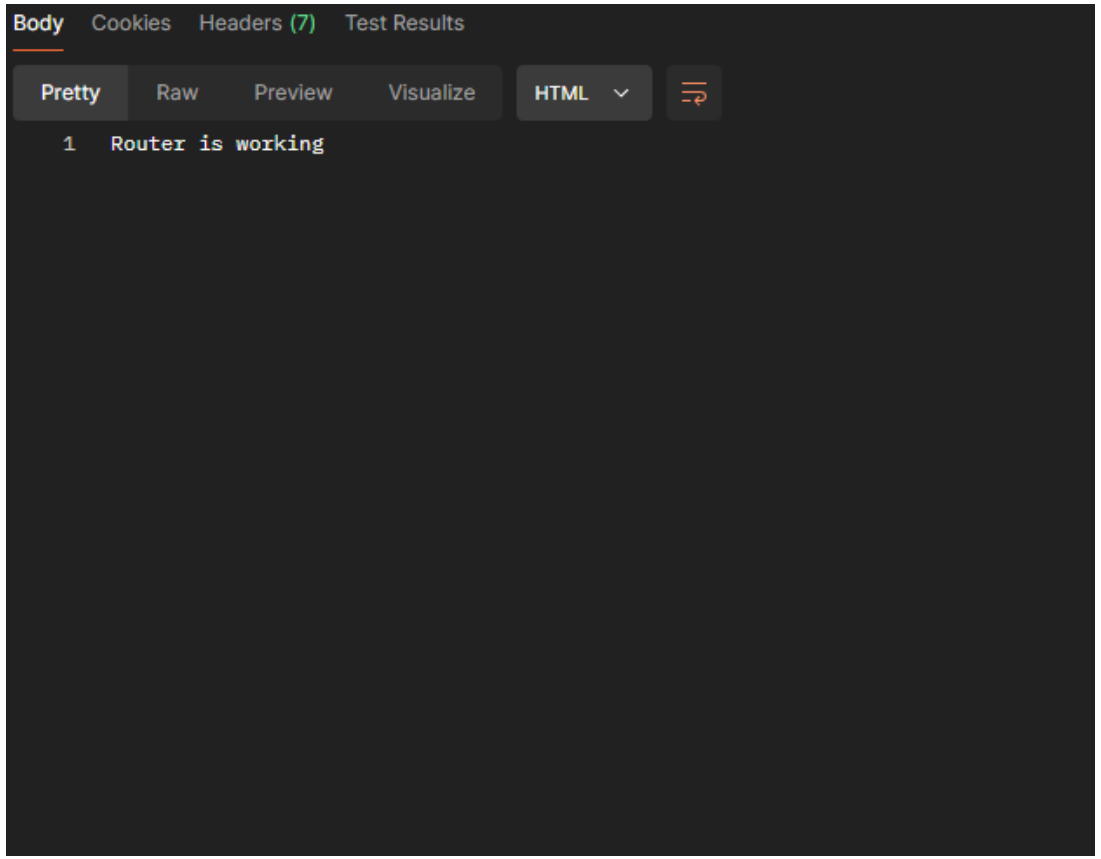

מחבר: גל לביא

Welcome/biz_cards_dev.rar

Router - תרגול

צרו תוכנית **Node.js** חדשה עבור הנושא ופתרו את התרגילים לפי הסדר.

תרגיל	תיאור המשימה
Ex-1	צרו נתב (Router) בשם "**router.js**" והגדירו בו נתיב עבור בקשות **GET** לשם בדיקת תפקודו.
Ex-2	הנתיב יחזיר אובייקט המכיל את התאריך הנוכחי והודעה המאשרת שהנתב תקין.
Ex-3	ייבאו את הנתב שייצרתם (**router.js**) אל קובץ ההרצה הראשי (**server.js**).
Ex-4	בדקו שבעת שליחת בקשת **GET** מתקבלת התשובה הצפויה.



ניהול Routing ב-Express.js – Request Params

בחלק זה נלמד כיצד ניתן לנהל תתי נתיבים ולדרוש פרמטרים בכניסה לנתיב.
בסיום הנושא תוכלו לענות על השאלות הבאות:

- כיצד נוכל לחלק את הנתיבים שלנו לתתי נתיבים לפי נושאים?
- כיצד נוכל להוסיף את הנתיבים המאוגדים בתתי הנתיבים לקובץ הניתוב הראשי (**router.js**)?
- מה הם **request params**?
- כיצד נוכל לדרוש פרמטר בכניסה לנתיב?
- כיצד נוכל להשתמש בפרמטר אשר נשלח מהמשתמש?
- מה זה נותן לנו?

```
router.post("/:name/:age", (req, res) => {
    const { name, age } = req.params;
    const user = {
        name,
        age
    };
    res.json({ message: "User was created", user });
});
```

POST localhost:8080/user/Eli/32

Body Cookies Headers (7) Test Results

Pretty Raw Preview Visualize JSON

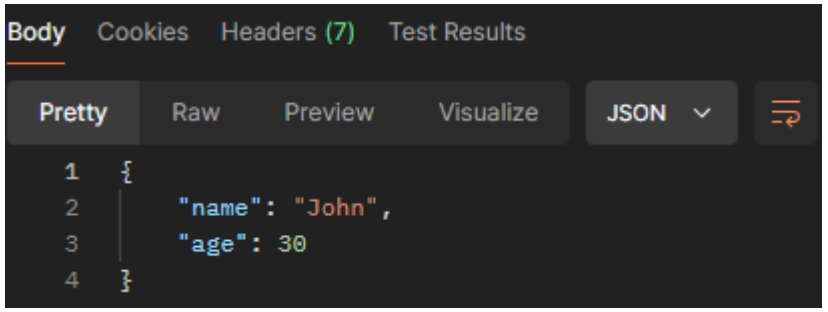
```
1 {
2     "message": "User was created",
3     "user": {
4         "name": "Eli",
5         "age": "32"
6     }
7 }
```

Welcome/biz_cards_dev.rar

User - תרגול – Request Params

המשיכו את התרגיל הקודם ופתרו את התרגילים לפי הסדר.

תרגיל	תיאור המשימה
Ex-1	צרו קובץ עבור תת נתיב בשם **user.routes.js**, צרו בו בקשת **GET**, ייבאו אותו לנתב הראשי (**router.js**) ובדקו שעובד.
Ex-2	צרו בקשת **GET** שתחזיר אובייקט עם שם וגיל של משתמש.
Ex-3	צרו בקשת **POST** שתדרוש שם וגיל כפרמטרים ותחזיר אובייקט עם שם וגיל תואמים למה שהמשתמש שלח.
Ex-4	צרו בקשת **PUT** שתדרוש שם וגיל כפרמטרים ותחזיר אובייקט עם השם והגיל של המשתמש לאחר עדכון.
Ex-5	צרו בקשת **DELETE** שתבקש סיסמא כפרמטר, אם הסיסמא אינה תואמת ל-"1234" יש להחזיר הודעת שגיאה עם סטטוס **403**, אחרת יש להחזיר הודעת הצלחה.



Welcome/biz_cards_dev.rar

Card - תרגול – Request Params

המשיכו את התרגיל הקודם ופתרו את התרגילים לפי הסדר.

תרגיל	תיאור המשימה
Ex-1	צרו קובץ עבור תת נתיב בשם **card.routes.js**, צרו בו בקשת **GET**, ייבאו אותו לנתב הראשי (**router.js**) ובדקו שעובד.
Ex-2	צרו בקשת **GET** שתחזיר אובייקט עם כותרת ותיאור של כרטיס ביקור.
Ex-3	צרו בקשת **POST** שתדרוש כותרת ותיאור כפרמטרים ותחזיר אובייקט עם כותרת ותאור תואמים למה שהמשתמש שלח.
Ex-4	צרו בקשת **PUT** שתדרוש תיאור כפרמטר ותחזיר אובייקט עם כותרת קבועה מראש והתאור שנשלח על ידי המשתמש.
Ex-5	צרו בקשת **PUT** שתדרוש כותרת כפרמטר ותחזיר אובייקט עם תיאור קבוע מראש והכותרת שנשלחה על ידי המשתמש.
Ex-6	צרו בקשת **DELETE** שתבקש סיסמא כפרמטר, אם הסיסמא אינה תואמת ל-"1234" יש להחזיר הודעת שגיאה עם סטטוס **403**, אחרת יש להחזיר הודעת הצלחה.

Body Cookies Headers (7) Test Results

Pretty Raw Preview Visualize JSON

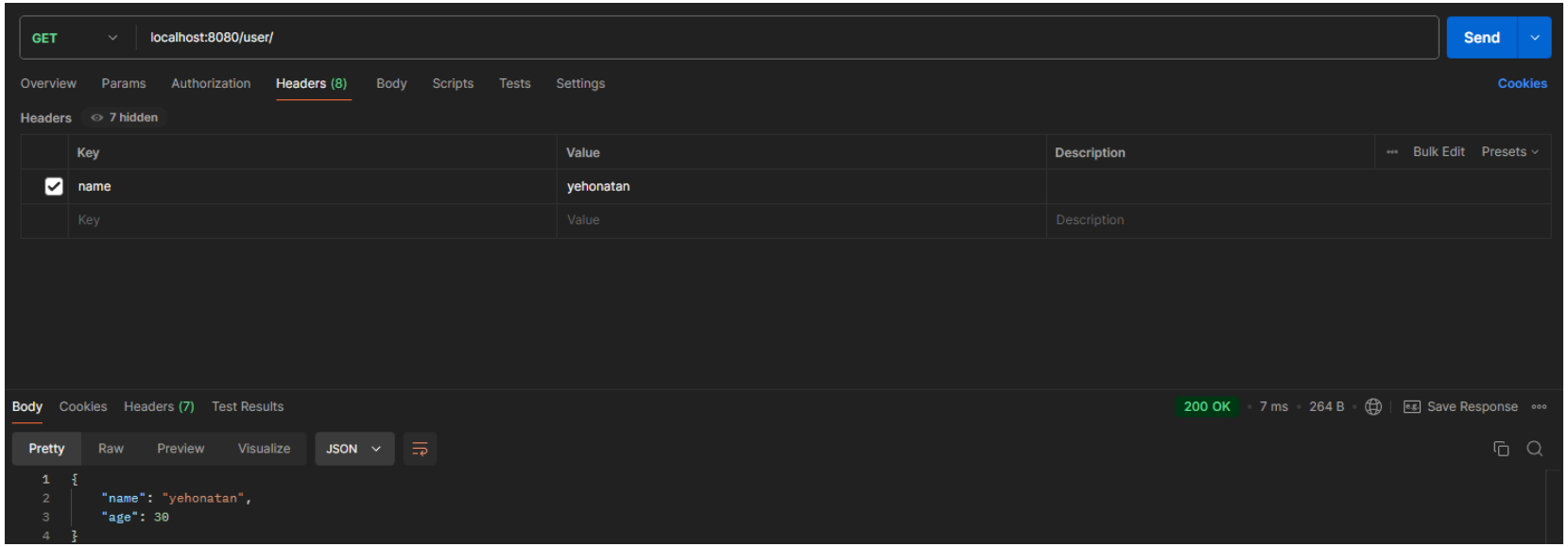
```
1 {
2     "title": "Card title",
3     "description": "Card description"
4 }
```

ניהול Routing ב-Express.js – Request Headers

בחלק זה נלמד כיצד ניתן לדרוש Headers בכניסה לנתיב.
בסיום הנושא תוכלו לענות על השאלות הבאות:

- מה כוללת בקשה (HTTP Request)?
- מה הם Headers?
- כמה Headers בקשה אחת יכולה להכיל?
- מה ניתן לשלוח ב-Headers?
- כיצד נוכל לבדוק אם header מסוים קיים בבקשה שהתקבלה לנתיב מסוים?
- מה זה נותן לנו?

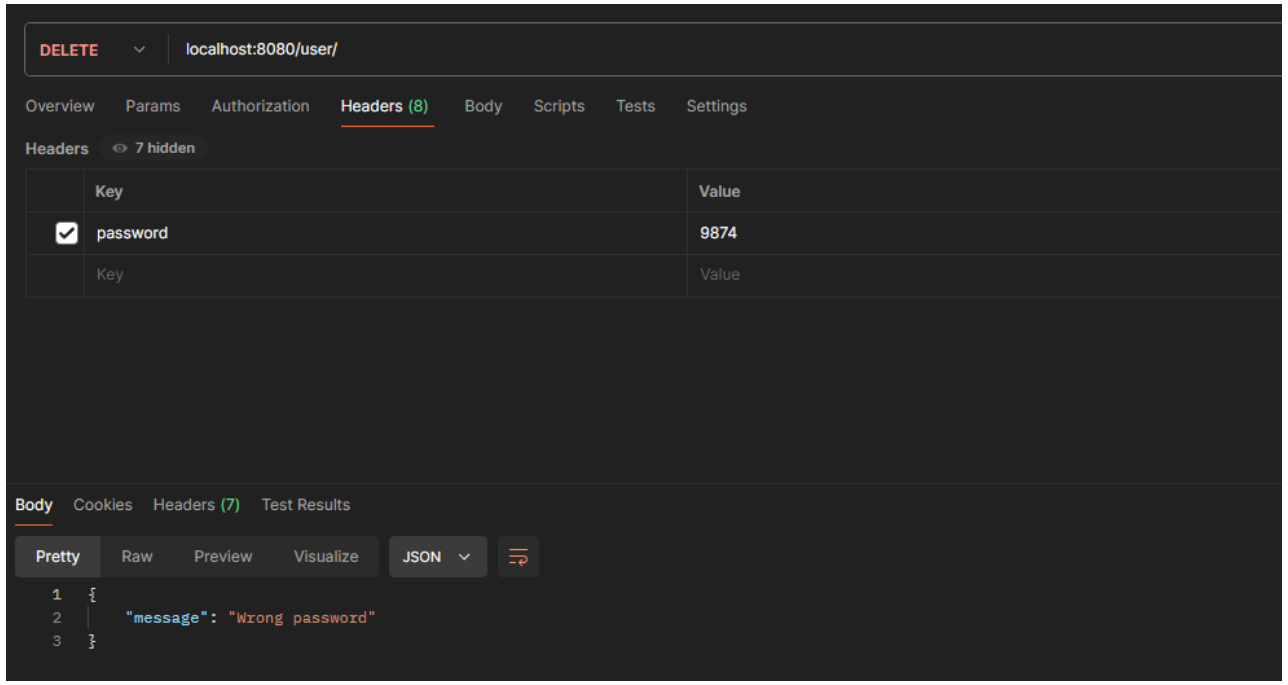
```
router.get("/", (req, res) => {
    const { name } = req.headers;
    const user = {
        name,
        age: 30
    };
    res.json(user);
});
```



Welcome/biz_cards_dev.rar

תרגול – Request Headers

המשיכו את התרגיל הקודם ופתרו את התרגילים לפי הסדר.



תרגיל	תיאור המשימה
Ex-1	שנו את בקשת ה-**GET** בנתיב "**user**" כך שהשם שיוחזר יתקבל ב-**headers**. במידה ולא קיים **header** כזה – השם שיוחזר באובייקט יהיה "**John**".
Ex-2	שנו את בקשת ה-**DELETE** בנתיב "**user**" כך שהסיסמא תתקבל ב-**headers**. במידה ולא קיים **header** כזה תוחזר שגיאה – אחרת – תוחזר הודעת הצלחה במידה והסיסמא תואמת ל-"**1234**", אם הסיסמא לא תואמת תוחזר הודעת שגיאה רלוונטית.
Ex-3	שנו את בקשת ה-**DELETE** בנתיב "**card**" כך שהסיסמא תתקבל ב-**headers**. במידה ולא קיים **header** כזה תוחזר שגיאה – אחרת – תוחזר הודעת הצלחה במידה והסיסמא תואמת ל-"**1234**", אם הסיסמא לא תואמת תוחזר הודעת שגיאה רלוונטית.

תודה על ההקשבה

אני וצוות המכללה כאן עבורכם