Node.js

מקורות מידע חיצוניים:

- גוגל AI - ג'ימיני
- מכללת האקריו - קורס בניית אתרים
- w3schools

מבוא:

Node.js היא סביבת ריצה (Runtime Environment) לקוד JavaScript שנכתבה ב-C++ ומבוססת על מנוע V8 של גוגל (אותו מנוע שמשמש בדפדפן כרום).

במילים פשוטות, Node.js מאפשרת לקוד JavaScript לרוץ מחוץ לדפדפן. עד הופעת Node.js, JavaScript שימשה בעיקר לפיתוח צד לקוח (Front-end), כלומר, קוד שרץ בתוך הדפדפן של המשתמש ומשפיע על האינטראקציה עם האתר. Node.js שינתה את זה לחלוטין ואפשרה ל-JavaScript לשמש גם לפיתוח צד שרת (Back-end), כלומר, קוד שרץ על השרת ומטפל בלוגיקה העסקית, בסיסי נתונים, תקשורת עם שירותים חיצוניים ועוד.

תכונות עיקריות של Node.js:

סביבת ריצה ל-JavaScript: מאפשרת להשתמש באותה שפה (JavaScript) הן בצד הלקוח והן בצד השרת, מה שמקל על מפתחי Full-stack ומפשט את תהליך הפיתוח.

מבוססת על מנוע V8: זהו מנוע JavaScript מהיר ויעיל מאוד, מה שהופך את Node.js לסביבת ריצה עם ביצועים גבוהים.

מודל I/O מונחה אירועים (Event-driven) ולא חוסם (Non-blocking): זוהי תכונה קריטית שהופכת את Node.js ליעילה במיוחד בטיפול במספר רב של בקשות במקביל. במקום לחכות שפעולה אחת תסתיים לפני שמתחילים פעולה אחרת (מודל חוסם), Node.js מאפשרת לפעולות אסינכרוניות להתבצע ברקע מבלי לחסום את הרצת הקוד הראשי. זה הופך אותה למתאימה ליישומים בזמן אמת כמו צ'אטים, סטרימינג ועוד.

Node.js :NPM (Node Package Manager) מגיעה עם מנהל חבילות מובנה ועשיר מאוד, המאפשר למפתחים לגשת לאלפי ספריות וכלים מוכנים לשימוש, מה שמזרז מאוד את תהליך הפיתוח.

חוצה פלטפורמות: ניתן להריץ אותה על מערכות הפעלה שונות כמו Windows, Linux ו-macOS.

שימושים נפוצים ב-Node.js:

- **פיתוח שרתי אינטרנט (Back-end):** בניית ממשקי API, שרתי HTTP, ויישומים מבוססי שרת.
- **יישומים בזמן אמת (Real-time applications):** כמו מערכות צ'אט, משחקי רשת, ויישומי שיתוף פעולה.
- **סטרימינג של נתונים:** יעילה מאוד בטיפול בזרמי נתונים גדולים בזמן אמת.
- **כלי שורת פקודה (Command-line tools):** כתיבת סקריפטים ואוטומציות.

מאור גרוס | פתרונות שיווק לעסקים

- **Microservices**: מתאימה לבניית ארכיטקטורת מיקרו-שירותים.

Node.js אינה שפת תכנות בפני עצמה, אלא **סביבת ריצה פורצת דרך** שאפשרה ל-**JavaScript**, שפה שהייתה מיועדת במקור לצד הלקוח, **להפוך לכלי רב עוצמה גם לפיתוח צד שרת ויישומים מורכבים.**

התקנת Node.js

לפני שמתחילים, ודא ש-Node.js מותקן על המחשב שלך.

הורדה: עבור לאתר הרשמי של Node.js

התקנה: הורד את הגרסה המומלצת (LTS - Long Term Support) עבור מערכת ההפעלה שלך (Windows, macOS, Linux) והתקן אותה. ההתקנה היא בדרך כלל רגילה " Next, Next, Finish". Node.js מגיעה יחד עם npm (Node Package Manager).

אימות ההתקנה:

פתח את ה-Terminal (ב-macOS/Linux) או ה-Command Prompt / PowerShell (ב-Windows) והקלד את הפקודות הבאות:

node -v

npm -v

*אם אתה רואה מספרי גרסה (לדוגמה: v20.12.2 עבור Node ו-10.5.0 עבור npm), סימן שההתקנה על המחשב שלך הצליחה.

פתיחת תיקייה ופרויקט חדש בVS Code-

פתח את **VS Code**.

פתח תיקייה חדשה: לך ל-File -> Open Folder...

צור תיקייה חדשה איפה שתרצה על המחשב שלך, לדוגמה: my-node-app.

בחר בתיקייה ופתח אותה.

יצירת קובץ Node.js בסיסי

יצירת קובץ: בתוך התיקייה שפתחת ב-VS Code, לחץ על אייקון "New File" (דף לבן עם סימן פלוס) בסרגל הצד השמאלי (Explorer), או לחץ קליק ימני ובחר New File.

שם הקובץ: קרא לקובץ app.js (או כל שם אחר עם סיומת js.

כתוב קוד Node.js פשוט: בקובץ app.js, כתוב את הקוד הבא:

```
// app.js

console.log("Hello from Node.js!");

// דוגמה פשוטה של שרת HTTP
const http = require('http'); // ייבוא מודול ה-HTTP המובנה של Node.js

const hostname = '127.0.0.1'; // localhost
```

מאור גרוס | פתרונות שיווק לעסקים

```
const port = 3000; // פורט שבו השרת יאזין

const server = http.createServer((req, res) => {
    res.statusCode = 200; // קוד סטטוס HTTP: OK
    res.setHeader('Content-Type', 'text/plain'); // סוג התוכן בתגובה
    res.end('Welcome to my Node.js server!\n'); // תוכן התגובה
});

server.listen(port, hostname, () => {
    console.log(`Server running at http://<span class="math-
inline">\{hostname\}\:</span>{port}/`);
});
```

<u>הרצת קוד Node.js מתוך VS Code</u>

ישנן שתי דרכים עיקריות להריץ את הקוד:

- **דרך הטרמינל המובנה של VS Code:** הדרך הנפוצה והמומלצת ביותר:
  - **פתח את הטרמינל:** ב-VS Code, לך ל-Terminal -> New Terminal (או השתמש בקיצור דרך: Ctrl + ~ ב-Windows/Linux, או Cmd + ~ ב-macOS).
  - **נווט לתיקיית הפרויקט**: ודא שאתה נמצא בתיקיית הפרויקט שלך (לדוגמה, my-node-app).
  - **הרץ את הקובץ:** הקלד את הפקודה הבאה:
    node app.js
    אתה אמור לראות את הפלט:
    Hello from Node.js!
    Server running at http://127.0.0.1:3000/
  - עכשיו פתח את הדפדפן שלך וגש לכתובת:
    http://localhost:3000/
    אתה אמור לראות את ההודעה Welcome to my Node.js server!.
    כדי לעצור את השרת, לחץ Ctrl + C בטרמינל.

- **הרצה באמצעות Debugger** (מומלץ לפיתוח מתקדם יותר):
  VS Code מגיע עם Debugger מובנה שתומך ב-Node.js, והוא מאוד שימושי לניפוי באגים.
  עבור ללשונית "Run and Debug": לחץ על אייקון החרק בצד שמאל בסרגל הפעילויות של VS Code (או Ctrl + Shift + D).
  צור קובץ launch.json: בפעם הראשונה, ייתכן שתתבקש ליצור קובץ launch.json. לחץ על הלינק create a launch.json file.
  בחר סביבה: בחר באפשרות Node.js. VS Code ייצור קובץ launch.json עם הגדרות ברירת מחדל.

```
{
    "version": "0.2.0",
        "configurations": [
            {
                "type": "node",
```

מאור גרוס | פתרונות שיווק לעסקים

```
                "request": "launch",
                "name": "Launch Program",
                "skipFiles": [
                    "<node_internals>/**"
                ],
                "program": "${workspaceFolder}\\app.js" //  ודא שזה מצביע
לקובץ ה-JS שלך
            }
        ]
}
```

ודא שהנתיב ל-"program" הוא "${workspaceFolder}\\app.js" (או שם הקובץ שלך).

הפעלת ה-Debugger:

חזור ללשונית "Run and Debug".

ודא שהאפשרות Launch Program נבחרה בתפריט הנפתח בראש החלונית.

לחץ על כפתור ה-**"Play"** הירוק.

הקוד ירוץ, ותוכל לראות את הפלט בחלונית "Debug Console" למטה.

היתרון כאן הוא שתוכל להגדיר נקודות עצירה (**Breakpoints**) בקוד על ידי לחיצה על השוליים ליד שורת קוד, ולעקוב אחר ערכי משתנים **בזמן אמת**.

שימוש ב-NPM (Node Package Manager)

NPM הוא כלי חיוני לכל פרויקט Node.js. הוא מאפשר לך לנהל תלויות (dependencies) – כלומר, ספריות קוד שנוצרו על ידי אחרים ואתה רוצה להשתמש בהן בפרויקט שלך.

<u>אתחול פרויקט Node.js:</u>

בטרמינל של VS Code (ודא שאתה בתוך תיקיית הפרויקט שלך), הקלד:

npm init -y

הפקודה הזו תיצור קובץ בשם package.json בתיקייה שלך. קובץ זה מכיל מטא-נתונים על הפרויקט שלך (שם, גרסה, תיאור) וחשוב מכך, הוא מנהל את רשימת התלויות (packages) של הפרויקט. הדגל -y מאשר את ברירות המחדל לשאלות ש-npm init היה שואל בדרך כלל.

<u>התקנה והסבר על חבילות חיצוניות</u> (לדוגמה, Express.js - framework פופולרי לבניית שרתי וב):

- **npm i express** - הוא **framework** (מסגרת עבודה) קטן וגמיש ליישומי אינטרנט של Node.js, שנועד ליצור יישומים מבוססי אינטרנט וממשקי API חזקים. זוהי למעשה שכבה דקה מעל ה-HTTP המובנה של Node.js.

למה משמשת בעיקר?

  - **בניית שרתי וב (Back-end):** זוהי הדרך הנפוצה ביותר לבנות שרתים שמקבלים בקשות HTTP (GET, POST, PUT, DELETE) ומחזירים תגובות (HTML, JSON, וכו').
  - **בניית RESTful APIs:** שימושית מאוד ליצירת ממשקי API עבור יישומי צד לקוח (Front-end) כמו React, Angular, Vue או יישומי מובייל.
  - **טיפול ב-Routing:** מאפשרת להגדיר בקלות נתיבים שונים (URLs) ולטפל בבקשות שמגיעות לכל נתיב.
  - **Middleware**: מציעה מנגנון חזק לטיפול בבקשות ותגובות בצורה מודולרית.

מאור גרוס | פתרונות שיווק לעסקים

- **npm install -g nodemon** - היא כלי עזר (utility). ה-g אומר שהיא מותקנת גלובלית במערכת, ולא רק בפרויקט הספציפי.

למה משמשת בעיקר?

  - **פיתוח מהיר (Hot Reloading):** תפקידה העיקרי הוא לנטר שינויים בקבצי המקור של Node.js. כאשר אתה שומר קובץ, Nodemon מזהה את השינוי ומפעילה מחדש אוטומטית את יישום ה-Node.js שלך. זה חוסך לך את הצורך לעצור ולהפעיל מחדש את השרת ידנית בכל פעם שאתה משנה קוד, ומזרז משמעותית את תהליך הפיתוח.
  - **חווית פיתוח משופרת:** מקל על איטרציות מהירות ובדיקות קוד בזמן פיתוח.

- **npm i mongoose** - היא ספריית Object Data Modeling (ODM) עבור MongoDB ו-Node.js. היא מספקת דרך מונחית אובייקטים לעבודה עם MongoDB.

למה משמשת בעיקר?

  - **אינטראקציה עם MongoDB**: אם אתה משתמש במסד נתונים NoSQL מסוג MongoDB (שאינו דורש סכימה קבועה), Mongoose הופכת את האינטראקציה איתו לקלה ונוחה יותר.
  - **אכיפת סכימה (Schema Enforcement):** למרות ש-MongoDB הוא NoSQL, Mongoose מאפשרת לך להגדיר "סכימות" (Schemas) שקובעות את המבנה והסוג של הנתונים שאתה מצפה לשמור בקולקציות שלך, מה שמסייע לשמור על עקביות הנתונים.
  - **אימות נתונים (Data Validation):** מספקת כלים לאימות נתונים לפני שמירתם במסד הנתונים.
  - **CRUD Operations**: מפשטת את פעולות ה- Create, Read, Update, Delete (CRUD) מול MongoDB.

- **npm i cors** - היא קיצור של **Cross-Origin Resource Sharing**. זוהי חבילת middleware פשוטה של Node.js (שבדרך כלל משמשת עם Express) המאפשרת לשרת שלך להתמודד עם בעיות אבטחה של מדיניות CORS.

למה משמשת בעיקר?

  - **פתרון בעיות אבטחה בין דומיינים:** כאשר יישום צד לקוח (לדוגמה, יישום React שרץ על http://localhost:**3001**) מנסה לבצע בקשות HTTP לשרת Node.js (לדוגמה, שרץ על http://localhost:**3000**), הדפדפנים **חוסמים** את הבקשות האלה מסיבות אבטחה (מדיניות "**Same-Origin Policy**").
  - **היתור גישה ממקורות שונים**: חבילת cors מאפשרת לך להגדיר בקלות אילו דומיינים (מקורות) מורשים לגשת למשאבים בשרת שלך, ובכך לפתור את חסימות ה-CORS מהדפדפן. חיונית כאשר יש לך Front-end ו-Back-end על כתובות שונות.

- **npm i jsonwebtoken** - היא ספרייה ליישום JWTs. **JSON Web Tokens** (**JWTs**) הם תקן פתוח (RFC 7519) שמשמש להעברת מידע באופן מאובטח בין צדדים בתור אובייקט JSON.

למה משמשת בעיקר?

  - **אימות משתמשים (Authentication):** הדרך המודרנית והפופולרית ביותר לאימות משתמשים ביישומי Web. במקום לשמור Session ID בשרת, השרת יוצר **JWT** לאחר התחברות מוצלחת, מחזיר אותו ללקוח, והלקוח שומר אותו ושולח אותו בכל בקשה עתידית. השרת מאמת את ה-JWT מבלי לגשת למסד נתונים.

מאור גרוס | פתרונות שיווק לעסקים

- הרשאה (**Authorization**): ניתן להכניס מידע על הרשאות המשתמש (לדוגמה, האם הוא אדמין) לתוך ה-JWT, והשרת יכול לבדוק זאת בכל בקשה.
- **העברת מידע מאובטחת**: משמשת להעברת מידע באופן שניתן לאמת את מקורו ואת שלמותו.

- **npm i lodash** - היא ספריית עזר (**utility library**) עשירה מאוד, שמספקת פונקציות עזר רבות לטיפול באובייקטים, מערכים, מספרים, מחרוזות, פונקציות ועוד ב-**JavaScript**.

למה משמשת בעיקר?

- **מניפולציה וטיפול בנתונים**: מפשטת משימות נפוצות ומסורבלות של טיפול בנתונים, כגון מיון מערכים, איטרציה על אובייקטים, מיזוג אובייקטים, בדיקת סוגים, סינון נתונים ועוד.
- **קיצור וייעול קוד**: במקום לכתוב לולאות ופונקציות מסורבלות לטיפול בנתונים, Lodash מספקת פונקציות קצרות, קריאות ויעילות שעושות את העבודה.
- **עקביות בין דפדפנים וסביבות**: מבטיחה שהפונקציות יתנהגו באופן עקבי בכל סביבות ה-JavaScript, גם עם גרסאות ישנות יותר של JavaScript.
- **דוגמאות לפונקציות**: _.map, _.filter, _.debounce, _.throttle, _.get, _.set, _.isEmpty, _.cloneDeep ועוד רבות.

<u>הפקודה הראשונה למשל (npm I express) תבצע את הדברים הבאים:</u>

- **תוריד** את חבילת express ותתקין אותה בתיקייה בשם node_modules בתוך הפרויקט שלך.
- **תוסיף** את express לרשימת התלויות בקובץ package.json שלך תחת dependencies.
- **אחרי** ההתקנה, תוכל להשתמש ב-Express בקובץ ה-JS שלך:

```
// server.js (או app.js)

const express = require('express'); // ייבוא חבילת Express
const app = express(); // יצירת יישום Express
const port = 3000;

app.get('/', (req, res) => {
    res.send('Hello from Express.js!');
});

app.listen(port, () => {
    console.log(`Express app listening at http://localhost:${port}`);
});
```

כדי להריץ את השרת החדש מבוסס Express, שמור את הקובץ (אם שינית את השם ל-server.js, אז node server.js בטרמינל).

<u>המלצות והרחבות:</u>

הרחבות ל-VS Code: ישנן הרחבות רבות שיכולות לשפר את חווית הפיתוח עם Node.js, לדוגמה:

מאור גרוס | פתרונות שיווק לעסקים

- **ESLint / Prettier**: לאימות קוד ועיצוב אוטומטי.
- **npm Intellisense**: להשלמה אוטומטית של מודולים ב-require()/import.
- **Path Intellisense**: להשלמה אוטומטית של נתיבים לקבצים.
- **מודולים מובנים:** Node.js מגיעה עם מודולים מובנים רבים (כמו http, fs - לטיפול בקבצים, path - לטיפול בנתיבים) ששווה להכיר.
- **תיעוד**: התיעוד הרשמי של Node.js הוא מקור מצוין למעטפת למידה רחבה ועדכנית.

מאור גרוס | פתרונות שיווק לעסקים