שיעור 10

מתחילים...


node
JS

הקדמה

NodeJS הינה סביבת עבודה להרצת JavaScript. סביבה זו בנויה בקוד פתוח למשתמש.
אחת מסביבות העבודה הנפוצות ביותר, מאפשרת הרצה של מגוון תחומים בתכנות (בעיקר עבודה בצד שרת).

למה NodeJS?

- מהירה, ומכילה הרחבות ותוספות רבות שניתן להשתמש בהן בקלות.
- זמני בקשה/תגובה מהירים יותר.
- יכולה לעבוד על אלפי בקשות מבלי להעמיס על המערכת.
- כותבים בשפת JavaScript
- מגוון רחב של ספריות שימושיות

הקדמה

NodeJS עובדת בעזרת המנוע V8.
V8 הינו המנוע שעליו פותח כרום (גוגל) לתרגום קבצי js לשפת מכונה.
מקימי node החליטו כבר ב2009 לעבוד עם מנוע זה.

ישנם מנועים נוספים להרצת קבצי js

Edge -> Chakra
Firefox -> SpiderMonky
Safari -> JavascriptCore

הקדמה

- יתרון נוסף (וחשוב מאוד) לעבודה ב NodeJS הוא - NodeJS עובדת בשיטה אסינכרונית.

(מטפלת בכמה בקשות בו זמנית)

NodeJS אסינכרוני

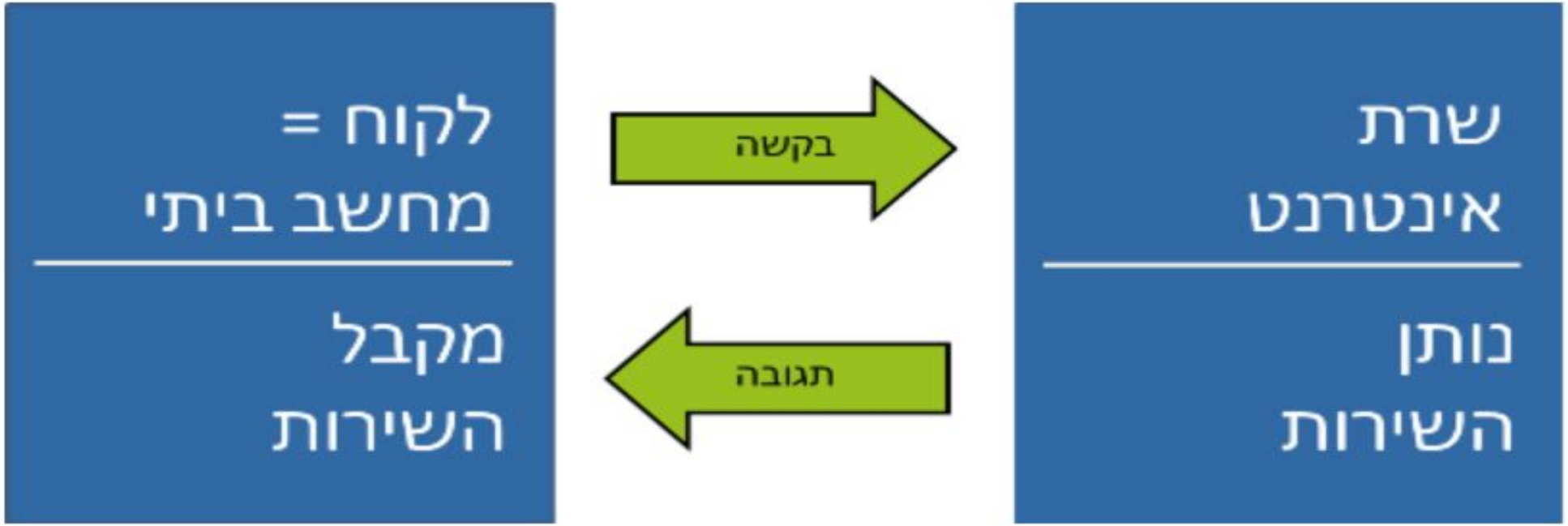
- מתקבלת בקשה
- שולח את המשימה למערכת
- מוכן לבקשה נוספת
- ברגע שהמערכת פתחה קראה את הקבצים והכינה את המידע
- מחזיר את המידע לclient

PHP/ASP.NET סינכרוני

- מתקבלת בקשה
- שולח את המשימה למערכת
- מחכה עד שהמערכת תפתח ותקרא את הקבצים ותכין את המידע
- מחזיר את המידע לclient
- מוכן לבקשה נוספת

nodeJS מריצה תכניות באופן אסינכרוני שגורם למערכת לעבוד בצורה חסכונית ויעילה יותר

איך עובד שרת ?


שרת
אינטרנט
נותן
השירות
בקשה
תגובה
לקוח =
מחשב ביתי
מקבל
השירות

התקנה

.ע"י כניסה לאתר הרשמי NodeJS ניתן להוריד

https://nodejs.org/en/

.Windows, Mac, Linux על כל מערכות ההפעלה הנפוצות NodeJS ניתן להפעיל את

הורדה של התוכנה תאפשר שימוש בNPM – Node Package Manager

ניתן לוודא שאכן node הותקן בהצלחה ע"י הרצת הפקודה בv– node CLI

Command Line Interface (CLI)

(מוכר גם כ CMD)

ממשק משתמש המסופק על ידי מערכת ההפעלה.

המשתמש יוכל להקיש פקודות ישירות למערכת ההפעלה, במידה והפקודה נכונה, המערכת תבצע אותה מיידית ואף יחזור חיווי (פלט) בהתאם.

שורת הפקודה

```
Microsoft Windows [Version 10.0.19043.1348]
(c) Microsoft Corporation. All rights reserved.

C:\Users\orgad>
```

יצירת פרויקט חדש בעזרת CLI

כעת נפתח את ה CLI ונגיע לתיקיה שבה ניצור את הפרויקט בעזרת הפקודות הבאות:

- dir – מציג את כל הקבצים הקיימים במיקום
- cd – פותח את התיקייה עם השם הרצוי (דוגמה: cd desktop)
- mkdir – יוצר תיקייה עם השם הרצוי (דוגמה: mkdir newFolder)
- type nul > - יוצר קובץ בפורמט והשם הרצוי (דוגמה: type nul > newFile.txt)
- node – מריץ את כל הפקודות JS שכתובות בקובץ (דוגמה: node index.js)

הרצת node.js

- הרצה של node מתבצעת באמצעות CLI.
- ניתן להריץ את node בעזרת הפקודה node . אופציה זו תאפשר לבצע פקודות ספציפיות מתוך הCLI.

```
$ node
> 4+4
8
> console.log('hello world');
hello world
```

- הרצת קבצי node nameOfFile.js – .js

```
$ node index.js
hello world
```

```
JS index.js ×
JS index.js
1  console.log('hello world');
```

OPM

הקדמה ל npm

- npm – Node Package Manager – ספריית מארזים סטנדרטית של Node.

- פותחה ב2010 על מנת לעזור לשתף "חבילות" מוכנות בין מתכנתים.
  כיום npm נחשבת למאגר ספריות הגדול ביותר בעולם – מכילה יותר מ11,500,000 חבילות קוד.

- "חבילה" – הינה קובץ/ספריה אשר מקונפגת ע"י הקובץ package.json.
  הגדרת כל חבילה יכולה להיות שייכת לארגון/משתמש פרטי וגם יכולה להיות ציבורית לקהל הרחב.

JSON
(JavaScript Object Notation)

- JSON הוא פורמט טקסטואלי (קריא) המיועד להעברת מקבצי מידע אשר מורכבים מזוגות key – value
- נוכל לזהות קבצים אלו ע"י הסיומת .json
- תחילה פורמט זה פותח לשימוש בjs, אבל כיום הוא לא תלוי שפה ונתמך על ידי מגוון שפות תכנות.
- מבני מידע המועברים בפורמט זה ניתנים לפענוח מהיר בjs ובדרך כלל קצר יותר.

```
{
  "Id": 0,
  "FirstName": "string",
  "LastName": "string",
  "Name": "string",
  "EmailAddress": "string",
  "TerritoryId": 0
}
```

פקודות CLI נוספות

- npm init יצירת חבילה חדשה
- npm install <package name> התקנת חבילה חדשה
- npm start הרצת חבילה

שלבי יצירת חבילה חדשה

1. npm init : פקודה ליצירת package json .

2. שדות חובה: שם חבילה וגרסה.

3. שדות שאינם חובה : תיאור, מחבר, נקודת התחלה , הגדרת סקריפטים נוספים והוספת חבילות חיצוניות.

```
{
  "name": "svcollege",
  "version": "1.0.0",
  "description": "just for example",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC"
}
```

הוספת חבילה חיצונית

כעת נבין איך מוסיפים חבילה חיצונית, כפי שציינו לפני כן ישנם המון חבילות מידע שנכתבו על ידי מתכנתים אחרים, בצורה זו נוכל להשתמש בפעולות פונקציונליות נפוצות למטרתנו.

פקודת התקנת חבילה חיצונית: npm install [module name]

```
>npm install superheroes
```

```
const superhero = require("superheroes");

var myHero = superhero.random();

console.log("hi, "+myHero);
```

Require(“name”) - פקודה המזמנת את החבילה לפרויקט

משימה

יש להיכנס לאתר https://www.npmjs.com/ המכיל את כל החבילות.
התקינו בעזרת הפקודות שלמדתם את שני החבילות הבאות:

1. superheroes
2. supervillains

צרו 2 שמות בכל פעם והדפיסו להדפיס ל console גיבור נגד נבל.

שימוש בחבילה קיימת

בתוך node ישנן המון חבילות בנויות שניתנות לשימוש לאחר שמזמנים אותן ל module שאנחנו כותבים.
לדוגמה: OS, HTTP, File System ועוד.

ניתן לראות אותן כאן https://nodejs.org/dist/latest-v12.x/docs/api/

בשביל להשתמש באותו module יש לזמן על ידי הפקודה require ולציין את שמה.

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

var totalmem = os.totalmem()
var freemem = os.freemem()

// Print OS memeory spaces in bit int
console.log("Total: " +totalmem);
console.log("Free: " +freemem);

var compOS = os.version()

// Print OS vertion
console.log("Path: "+compOS);
```

נקודת התחלה

- הרצת חבילה תתבצע בעזרת מספר הגדרות פשוטות.
- יצירת קובץ index.js קובץ זה מוגדר כנקודת התחלה בpackage.json .
- הגדרת מאפיין (key) start תחת קובץ קונפיגורציה.
- הגדרת הערך (value) - node filename.js.
- הרצת הפקודה npm start..

```
{
  "name": "svcollege",
  "version": "1.0.0",
  "description": "just for example",
  "main": "index.js",
  "scripts": {
    "start": "node index.js",
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC"
}
```

```
$ npm start

> svcollege@1.0.0 start C:\Users\ella\Desktop\קמ יוביג\svcollege\BackEnd\node example
> index.js
```

Import & Export

require ('nameOfmodule') ייבוא של ספרייה .

ייבוא מקבל רק את הערך הספציפי שהתקבל מהספרייה.

module.exportsייצוא ערך/פונקציה.

דוגמא ליצירת ספרייה sum :

```
sum = (n1,n2)=>(n1+n2);
module.exports = sum;
```

בדוגמא זו נבנה module אשר מקבל 2 ערכים ומחזיר את הסכום של שניהם.
בנוסף נציג ייבוא של ספרייה ושימוש שלה.

יצירת הsum .

ייבוא הsum מהmodule .

```
const sum = require('./sum.js');
res = sum(1,1);
console.log(res);
```

משימה

יש ליצור ספריית מחשבון 4 פעולות אשר מייצא את כל הפעולות .

יש לייבא את הספרייה לתוך הקובץ הראשי שלנו ולהשתמש ב4 הפעולות, כאשר את התוצאה מדפיסים לconsole.

File System

File System

fs – הינה ספרייה לעבודה מול קבצים.
ספרייה זו מאפשרת ליצור קבצים, למחוק קבצים, לכתוב לתוך קובץ ולקרוא מהקבצים.

<u>פתיחת קובץ</u>

1. קריאה לספרייה fs.
2. פונקצייתopen מקבלת 3 ערכים.
   א נתיב לקובץ.
   ב סוג הקובץ 'w' קריאה, כתיבה וכו'.
   ג. callBackFunction אשר זורק שגיאה במידת הצורך.

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

fs.open('file.txt', 'w', (err) => {
    if (err) throw err;
  });
```

File System - Write

כתיבה לקובץ פונקציית קריאה לקובץ מאפשרת העברת מידע מהקוד שלנו לתוך הקובץ הקיים.

** במידה והקובץ לא קיים היא יוצרת קובץ קיים.

** במידה והוא לא ניתן לשינוי היא תזרוק הודעת שגיאה.

writeFile פונקציה זו מקבלת 3 ערכים :

1. נתיב הקובץ.
2. DATA המידע אשר יכנס לקובץ.
3. callBackFunction לזריקת שגיאות.

```
fs.writeFile('file.txt','hello world',(err)=>{
    if (err) throw err;
});
```

File System - Append

<u>צירוף לקובץ</u> פונקציית קריאה לקובץ מאפשרת העברת מידע מהקוד שלנו לסוף של קובץ קיים
** במידה והקובץ לא קיים היא יוצרת קובץ קיים.

** במידה והוא לא ניתן לשינוי היא תזרוק הודעת שגיאה.

<u>appendFile</u> – פונקציה זו מקבלת 3 ערכים
1. נתיב הקובץ
2. DATA המידע אשר יכנס לסוף הקובץ
3. callBackFunction לזריקת שגיאות.

```
fs.appendFile('file.txt','append text',(err)=>{
  if (err) throw err;
});
```

משימה

i. צרו קובץ חדש בשם "aboutMe.txt" לכתיבה "w"

i. כתבו לתוך הקובץ את השם המלא שלכם

i. עדכנו את הקובץ והוסיפו לשם המלא את העיר שבה אתם גרים

File System - Read

<u>קריאה מקובץ</u> פונקציה אשר מאפשרת לקרוא ערכים מקובץ ולהחזיר אותם למשתנה.

<u>ReadFile</u> פונקציה זו מקבלת 2 ערכים:

1. נתיב הקובץ.
2. callBackFunction מקבלת 2 ערכים, שגיאה ואת ערכי הקובץ.

```
fs.readFile('file.txt',(err,data)=>{
    if (err) throw 'not';
    console.log(data.toString());
});
```

בדוגמא זו הערך מהקובץ מודפס בconsole.

File System – More Options

<u>מחיקת קובץ</u> פונקציה אשר מוחקת קובץ ספציפי

<u>Unlink</u> מקבלת 2 ערכים:
1. שם הקובץ.
2. callBackFunction

```
fs.unlink('file.txt',(err)=>{
    if(err)throw err;
})
```

<u>משנה שם של קובץ</u> פונקציה אשר משנה את שם הקובץ.

<u>rename</u> – פונקציה אשר מקבלת 3 ערכים:
1. נתיב הקובץ.
2. שם חדש לקובץ.
3. callBackFunction מקבלת 2 ערכים (שגיאה ואת ערכי הקובץ).

```
fs.rename('file.txt', 'changeName.txt',(err)=>{
    if (err) throw err;
  });
```

משימה

יש לקרוא מתוך קובץ טקסט ולהדפיס את כמות המילים המופיעה בו, נקודת הנחה שבין כל מילה יש רווח אחד בלבד.

סוף שיעור 10

האם יש שאלות?