svcollege
ללמוד. לדעת. לעבוד.

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

svcollege
ללמוד. לדעת. לעבוד.

הקדמה

- יתרון נוסף (וחשוב מאוד) לעבודה ב NodeJS הוא - NodeJS עובדת בשיטה <u>אסינכרונית</u>.

(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

svcollege
ללמוד. לדעת. לעבוד.

התקנה

ניתן להוריד NodeJS ע"י כניסה לאתר הרשמי.
https://nodejs.org/en/

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

הורדה של התוכנה תאפשר שימוש בNPM – Node Package Manager.

ניתן לוודא שאכן node הותקן בהצלחה ע"י הרצת הפקודה ב CLI ← node –v.
(תציג לנו את הגרסא של node).

svcollege
ללמוד. לדעת. לעבוד.

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>
```

svcollege
ללמוד. לדעת. לעבוד.

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)

svcollege
ללמוד. לדעת. לעבוד.

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 js – node nameOfFile.js.

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

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

שלום

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.

svcollege
ללמוד. לדעת. לעבוד.

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

svcollege
ללמוד. לדעת. לעבוד.

שלבי יצירת חבילה חדשה

1. npm init – פקודה ליצירת package
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

שימוש בחבילה קיימת

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

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

```
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);
```

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

נקודת התחלה -

- הרצת חבילה תתבצע בעזרת מספר הגדרות פשוטות.
- יצירת קובץ index.js קובץ זה מוגדר כנקודת התחלה בpackage.json.
- הגדרת מאפיין start (key) תחת הקובץ קונפיגורציה.
- הדגרת 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
```

svcollege
ללמוד. לדעת. לעבוד.

Import & Export

require('path') – ייבוא של ספרייה
ייבוא מקבל רק את הערך הספציפי שהתקבל מהספרייה

module.exports - ייצוא ערך/פונקציה

דוגמא ליצירת ספרייה sum.

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

יצירת הsum

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

ייבוא ה sum מהmodule

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

svcollege
ללמוד. לדעת. לעבוד.

משימה

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

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

File System

svcollege
ללמוד. לדעת. לעבוד.

File System

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

<u>פתיחת קובץ –</u>

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

```
1  const fs = require('fs');
2
3  fs.open('file.txt', 'w', (err) => {
4      if (err) throw err;
5    });
```

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;
});
```

svcollege
ללמוד. לדעת. לעבוד.

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;
});
```

svcollege
ללמוד. לדעת. לעבוד.

svcollege
ללמוד. לדעת. לעבוד.

משימה

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

ii. כתבו לתוך הקובץ את השם המלא שלכם

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

File System - Read

קריאה מקובץ - פונקציה אשר מאפשרת לקרוא ערכים מקובץ ולהחזיר אותם למשתנה.

ReadFile – 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.

svcollege
ללמוד. לדעת. לעבוד.

File System – More Options

מחיקת קובץ- פונקציה אשר מוחקת קובץ ספציפי

Unlink – מקבלת 2 ערכים.
1. שם הקובץ.
2. callBackFunction

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

משנה שם של קובץ– פונקציה אשר משנה את שם הקובץ..

rename – 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;
  });
```

svcollege
ללמוד. לדעת. לעבוד.

svcollege
ללמוד. לדעת. לעבוד.

Express

http://

HTTP

Hypertext Transfer Protocol

svcollege
ללמוד. לדעת. לעבוד.

URL

גישה לכל דף באינטרנט תיעשה על ידי שורה אחת הנקראת URL או נתיב. בעזרת נתיב זה נבין כמה נקודות חשובות של התקשורת בין שרת ללקוח.
אותו URL מורכב מכמה חלקים, ולכל חלק חשיבות משלו בתקשורת.

https://www.ynet.co.il/food

Secure Protocol
אוסף כללים לתקשורת בין השרת ללקוח

App domain
שם השרת המארח

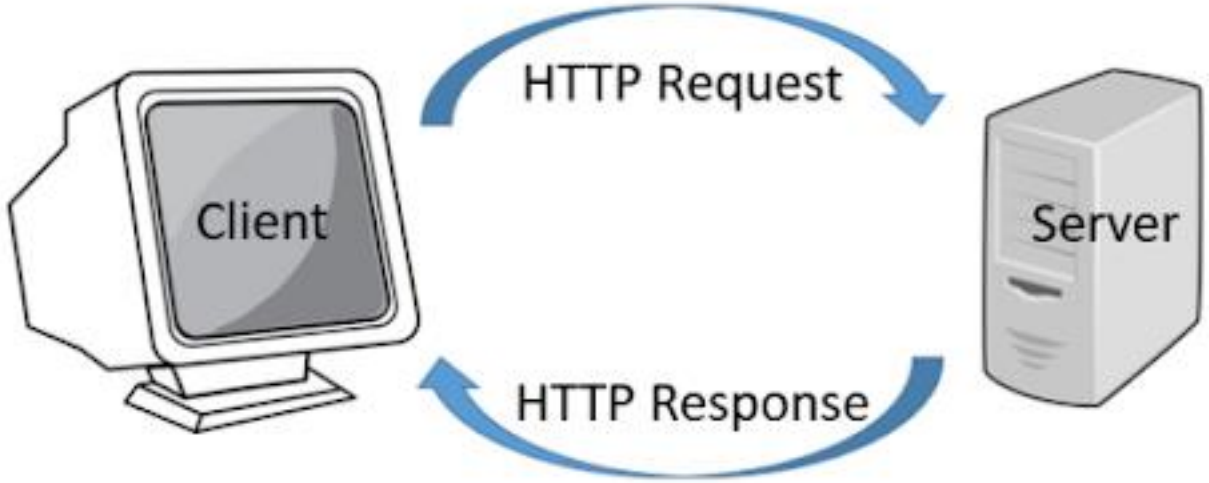
Client Request
בקשת הלקוח לדף אינטרנט מסוים

הקדמה

http://

מהו המושג HTTP?

הוא פרוטוקול אינטרנט שנועד עבור העברת מידע כגון דפי html, css, javascript ואלמנטים נוספים המשמשים להצגת אתר ברשת. התקשורת המתבצעת היא בין שתי נקודות שהן client ו server. ישנו דיאלוג המתקיים בין שניהם הנקרא בקשה ותגובה לדוגמה: כאשר המשתמש מקליק על קישור באתר השרת מחזיר את המידע הרלוונטי והדפדפן בונה את האתר בעזרת אותו מידע.



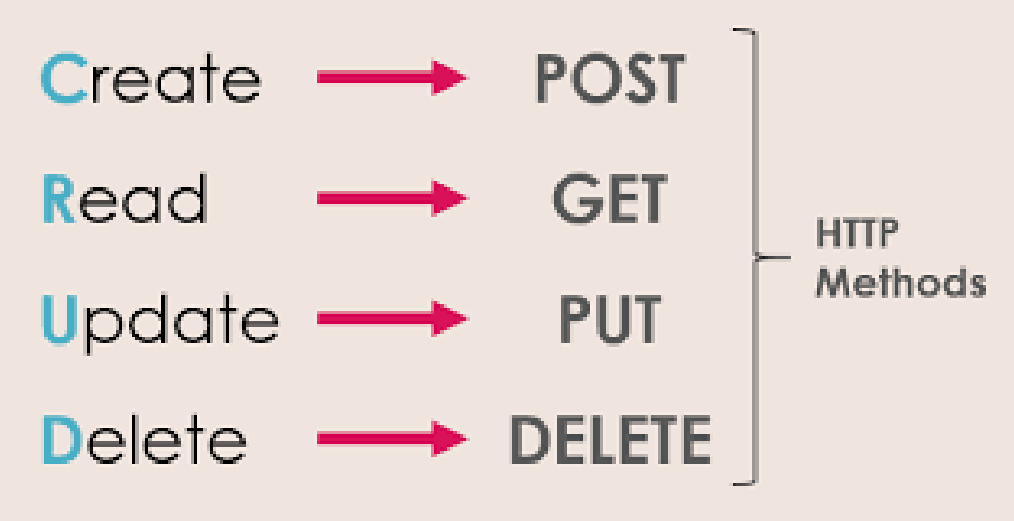
svcollege
ללמוד. לדעת. לעבוד.

REST API - Representational state transfer

בכדי ליצור שרת אינטרנט תחילה נבין אילו שירותים השרת מציע ואיך כל פעולה/שיטה עובדת.

ניישם מספר עקרונות עיקריים – יצירה, קריאה, עדכון ומחיקה.

עקרונות אלו נקראים CRUD operations.



REST API - Representational state transfer

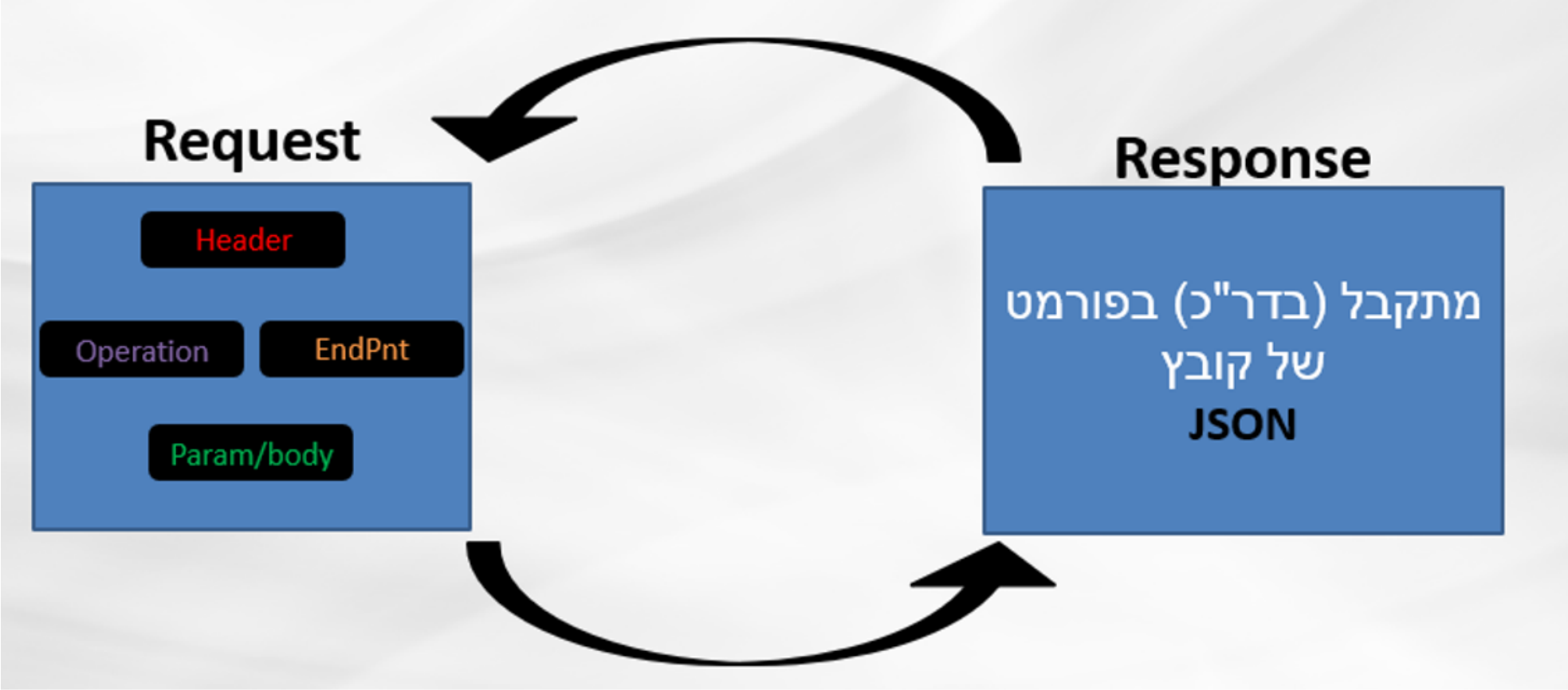
איך נראה בקשה (Request) שנשלחת לשרת?

מורכבת מ 4 חלקים מרכזיים:

- **Header** - חלק מיוחד בבקשות API מכיל key או מידע לאימות הבקשה.
- **Operation** - מכיל אחת מפעולות הcrud.
- **Endpoint** - מיקום/ניתוב המידע השמור בתוך השרת.
- **Param/body** - מידע שנרצה לשלוח בתוך הבקשה עצמה.

svcollege
ללמוד. לדעת. לעבוד.

REST API - Representational state transfer



svcollege
ללמוד. לדעת. לעבוד.

Express Package

npm

svcollege
ללמוד. לדעת. לעבוד.

Express

Express – הינה ספרייה סטנדרטית ליצירת אפליקציות ווב.
ספרייה זו הינה אחת הספריות הפופולריות לבניית שרת אשר תומך באפליקציות ווב.

```
"dependencies": {
  "express": "^4.17.1"
}
```

התקנה –
במסך ה CLI –

npm install express –save . יוסיף dependencies בקובץ קונפיגורציה.

```
const express = require('express');
const app = express();
const port = 3000;

app.get('/', (req, res) => res.send('Hello World'));

app.listen(port, () => console.log(`listening to port ${port}`));
```

דוגמא לבניית שרת http אשר מאזין לפורט 3000
ומציג hello world

Express – build server

const express = require('express'); הוספת הספרייה express. ** אין צורך בנתיב רק בשם הספריה

const app = express(); ייבוא האובייקט למשתנה app.

app.get('/',(req,res) => res.send('hello world')); הפעלת פונקצית GET, מקבלת ROUT ופונקציה אשר
מקבלת בקשה ותגובה.
מחזירה הודעה Hello world.
*פעולת GET "מבקשת" מידע ממקור ספציפי.

app.listen(3000 , () => console.log('listen to port 3000')); האזנה של השרת לפורט ספציפי. ומחזירה הודעה בlog.

Express – static files

server

במידה ונרצה להריץ קבצים סטטים מהשרת כמו – קבצי HTML , CSS , Imaes וכו'.
נוכל להגדיר בעזרת הפקודה use ערוץ ספציפי שבו הקובץ יוכל לרוץ.

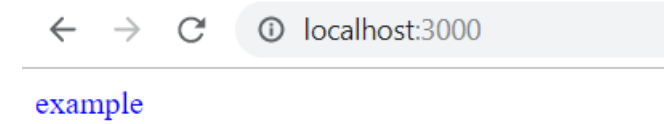
```
app.use('/',express.static('html'));
```

'/' – נתיב יחסי
'html' – שם תיקייה

Html file

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Document</title>
</head>
<body>
    <div id="" style="color: blue">example</div>
</body>
</html>
```

browser



svcollege
ללמוד. לדעת. לעבוד.

svcollege
ללמוד. לדעת. לעבוד.

משימה

#1 יש ליצור 3 עמודי html
העמוד הראשון יתאר את עמוד הבית.
העמוד השני יתאר עמוד רישום
והשלישי עמוד אודות.

יש להגדיר ערוץ לכל אחד מהעמודים בנפרד.

Express – body-parser

Body-parser – הינה ספרייה אשר נועדה לעבוד מול דפי HTML ולהעביר מידע דרך POST METHOD.
את המידע ניתן להעביר דרך קבצי JSON , URL , Buffer , String.

על מנת להשתמש בספרייה זו ראשית נצטרך להתקין אותה על גבי הפרויקט
– CLI
npm I body-parser –save

```
var express = require('express');
var app = express();

var bodyParser = require("body-parser");
app.use(bodyParser.urlencoded({ extended: false }));
```

לאחר ההתקנה נצטרך להוסיף את האוביקט לפרויקט שלנו.
ולהגדיר את הדרך שבה היא תעביר נתונים.

svcollege
ללמוד. לדעת. לעבוד.

Express – Form Example

בדוגמא זו אנחנו נציג דוגמא ליצירת עמוד פורום אשר יפתח בעליית השרת. ולאחר הזנת הנתונים הפרטים יועברו לשרת בPOST ויועברו לערוץ אחר שם אנחנו נציג את פרטי הלקוח שנכנס.

Index.html

```html
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
</head>
<body>
    <form action="/submit-student-data" method="post">
        First Name: <input name="firstName" type="text" /> <br />
        Last Name: <input name="lastName" type="text" /> <br />
        <input type="submit" />
    </form>
</body>
</html>
```

יצירת פורום
בעל 2 שדות טקסט
וכפתור submit

* שימו לב לפעולה אשר מחזירה
את הכתובת URL
ואת ההכרזה על POST

svcollege
ללמוד. לדעת. לעבוד.

Express – Form Example

```
1   var express = require('express');
2   var app = express();
3   var bodyParser = require("body-parser");
4   app.use(bodyParser.urlencoded({ extended: false }));
5   app.get('/', function (req, res) {
6       res.sendFile(__dirname + '/index.html');
7   });
8   app.post('/submit-student-data', function (req, res) {
9       var name = req.body.firstName + ' ' + req.body.lastName;
10
11      res.send(name + ' Submitted Successfully!');
12  });
13  app.listen(3000, function () {
14      console.log('Node server is running..');
15  });
```

3: משיכת הספרייה
4: הכרזה על משיכה מהURL
5: פעולת GET והחזרה של קובץ HTML
8: פעולת POST מאזינה לפורום שיחזיר בקשה ומציגה את ההחזר שלו על העמוד הבא.
13: הפעלת השרת לפורט 3000

svcollege
ללמוד. לדעת. לעבוד.

DATABASE

NoSQL

svcollege
ללמוד. לדעת. לעבוד.

Database

בסיס נתונים (מאגר מידע) – הינו מקום מסודר לאחסון נתוני מחשב, בסיס נתונים נשמר לרוב בזיכרון הקשיח של המחשב או על גבי שרתים ייעודיים. בסיס נתונים מאפשר גישה מלאה לשמירה, עדכון ושליפת המידע. ישנם כמה מודלים לעבודה מול בסיס נתונים.

svcollege
ללמוד. לדעת. לעבוד.

NoSQL

מסד נתונים בעל נפחים גדולים, השומר מידע בכמויות עצומות, הנכנס במהירות גבוהה מאוד. מאגר מידע שמאפשר גישה למידע שאינו בנוי כמבנה טבלאי בעלי יחסים. מידע זה אינו מאורגן לפי שיטה כלשהי, הוא מגוון מאוד כך שלא ניתן לסדר אותו בטבלאות.

מהו ההבדל?

עבודת השרת מול SQL

עבודת השרת מול NoSQL

svcollege
ללמוד. לדעת. לעבוד.

MongoDB

MongoDB – הינו מסד נתונים העובד מול מסדי נתונים NoSQL .

הורדת התוכנה www.mongodb.com.
יש להתקין את התוכנה הרצויה.

בעת ההתקנה יש להוריד את הסימון מ install mongoDBcompass.

svcollege
ללמוד. לדעת. לעבוד.

NoSQL

NoSQL – הינו מסד נתונים לא רלציוני. אשר שומר את כל הנתונים שלו תחת collections בניגוד לטבלאי ששומר הכל בטבלאות.

```
[
    {
        "id":"b032154",
        "firstName":"shem",
        "lastName":"bar",
        "email":"shem@svcollege.co.il",
        "Title":"Manager"
    },
    {
        "id":"b032155",
        "firstName":"dor",
        "lastName":"chen",
        "email":"dor@svcollege.co.il",
        "Title":"Teacher"
    },
    {
        "id":"b032156",
        "firstName":"max",
        "lastName":"nudler",
        "email":"max@svcollege.co.il",
        "Title":"Teacher"
    }
]
```

Collection – הינו מערך של אובייקטים

Document – אובייקט אשר מכיל מאפיינים וערכים

Props – מאפיין (כותרת לערך) – מכיל ערך

Value – הערך עצמו.

svcollege
ללמוד. לדעת. לעבוד.

Create

show dbs

מציג את כל המאגרי המידע בשרת.

use dbName

מייצר DB חדש (במידה וקיים אז פשוט ניגשת אליו).

db

מחזיר את השם של ה db שאיתו אנחנו עובדים.

db.dropDatabase()

מחיקת כל מאגר המידע

Insert

svcollege
ללמוד. לדעת. לעבוד.

db.collectionName.insertOne({ id: 1, name: ”Bamba”, price: 5 })

פקודה זו יוצרת collection בשם nameOfCollection
ומכניסה לתוכו document חדש עם הערכים

insertMany

ניתן להוסיף כמה ערכים שנרצה לאותו collection.

show collections

מציג את כל הcollections השמורים במאגר המידע

db.collectionName.drop()

מחיקת ה collection מתוך מאגר המידע

תרגול

צרו מאגר מידע חדש בשם השכרת רכבים CarRental_db

צרו collection חדש בשם avCars (רכבים פנויים)
והוסיפו מערך אובייקטים הכולל לפחות 3 רכבים פנויים להשכרה.
כל אובייקט יכיל את המאפיינים הבאים:

- מספר רכב
- דגם
- שנתון
- צבע

Read

svcollege
ללמוד. לדעת. לעבוד.

db.collectionName.find()

תציג את כל האובייקטים של ה collection

פקודת find, מקבלת 2 ערכים אופציונליים בעזרתם נוכל לסנן את המידע ולהציג את המידע המתאים.

- **query** – מסנן את כל הערכים אשר לא עומדים בתנאי
- **projection** – מציין אילו שדות יש להחזיר בהנחה שעומדים בתנאי

db.collectionName.find({ filed: {operator: value}}, { filed: 1/0 })

תרגול

- הציגו את הרכבים בצבע שחור בלבד
- הציגו את הרכבים החדשים ביותר (שנתון 2019 לפחות)

svcollege
ללמוד. לדעת. לעבוד.

Update

db.collectionName.update()

תעדכן את האובייקט בתוך ה collection על פי תנאי מסוים
ניתן לעדכן גם מספר אובייקטים העונים לאותו תנאי

פקודת זו תקבל ותנאי ומאפיין להוספת לתוך האובייקט

- **תנאי** – לסינון עבור האובייקט המתאים ב collection
- **מאפיין וערך** – הוספת מאפיין חדש עם וערך לאותו אובייקט

```
db.collectionName.updateOne({ filed: value }, { $set: { filed: value}})
```

NoSQL

svcollege
ללמוד. לדעת. לעבוד.

db.collectionName.deleteMany()

תמחק את האובייקט/ים מתוך ה collection על פי תנאי מסוים

פקודת זו תקבל ותנאי

- **תנאי** – לסינון עבור האובייקט המתאים ב collection

```
db.collectionName.deleteMany({ filed: value})
```

mongoose
npm package Install

1 התקנת הספרייה

```
$ npm install mongoose
```

2 יבוא הספרייה לפרויקט ויצירת חיבור לשרת

```
const mongoose = require('mongoose');
mongoose.connect('mongodb://localhost/test', {useNewUrlParser: true});
```

mongoose
Connect verification

svcollege
ללמוד. לדעת. לעבוד.

כעת יש לנו חיבור בהמתנה לבסיס הנתונים, עלינו לקבל הודעה אם אנו מתחברים בהצלחה או אם מתרחשת שגיאת חיבור כלשהי.

3 בדיקת החיבור לבסיס הנתונים

```
const db = mongoose.connection;
db.on('error', console.error.bind(console, 'connection error:'));
db.once('open', function() {
  console.log("Connected!")
});
```

mongoose
Schema and model

svcollege
ללמוד. לדעת. לעבוד.

4 יצירת Schema

הינה מחלקה אשר נועדה לקביעת
תבנית ה Collection

א ראשית יש ליצור הפנייה למחלקה.

ב שנית נכתוב את המבנה לשמירת המידע

```
const productSchema  = mongoose.Schema({
    name: String,
    price: Number,
    stock: Number
});
```

5 יצירת מודל

```
const Product = mongoose.model("Product", productSchema);
```

mongoose
new document

svcollege
ללמוד. לדעת. לעבוד.

```
const toblerone = new Product({
    name: "Toblerone",
    price: 10,
    stock: 50
});
```

6 יצירת document

א ראשית יש ליצור הפנייה למודל.

ב שנית נזין את המידע על פי המבנה

```
toblerone.save()
```

7 שמירת ה document לתוך ה collection

8 בדיקת הערכים במאגר – db.products.find()

```
{ "_id" : ObjectId("5f57e2e38d171c04fe11e486"), "name" : "Toblerone", "price" : 10, "stock" : 50, "__v" : 0 }
```

mongoose find

- שליפת collection מתוך המאגר

```
Product.find((err, products)=>{
    if(err) throw err
    else console.log(products)
});
```

- שליפה והצגה של מאפיין מסוים מתוך המאגר

```
Product.find((err, products)=>{
    if(err) throw err
    else {
        products.forEach((product)=>{
            console.log(product.name)
        })
    }
});
```

mongoose
Update & Delete

svcollege
ללמוד. לדעת. לעבוד.

- עדכון מאפיין ב document

```
Product.updateOne({ _id: "5f57fcc6e299ae0d5b6fd5bf" }, { name: "Loacker "}, (err)=>{
    if(err) throw err
    else console.log("Data updated");
});
```

- מחיקת document מה collection

```
Product.deleteOne({ name: "Loacker "}, (err)=>{
    if(err) throw err
    else console.log("Data deleted");
});
```

mongoose
Update & Delete

- יצירת קשרים בין documents
  - יצירת schema חדשה המאפיינת קשר מ prop אל document

a

```
const reviewSchema = mongoose.Schema({
    review: String,
    rating: Number
});

const Review = mongoose.model("Review", reviewSchema);

const review = new Review({
    review: "Sweet candy!",
    rating: 8
});

review.save()
```

b

```
const productSchema  = mongoose.Schema({
    name: String,
    price: Number,
    stock: Number,
    review: reviewSchema
});
```

c

```
const snickers = new Product({
    name: "Snickers",
    price: 6.5,
    stock: 72,
    review: userReview
});

snickers.save()
```