**🧠 סיכום מורחב – שיעורים 9–11: Node.js, Express, HTTP, REST API**

**🧑‍🏫 שיעור 9 – Node.js וסביבת הפיתוח בצד השרת**

**🔧 מה זה Node.js?**

* סביבת הרצה להרצת JavaScript **מחוץ לדפדפן** (בצד השרת).
* מאפשרת בניית שרתים ו־APIs.
* קוד פתוח, מהיר, תומך באסינכרוניות.

**💻 קוד פתוח (Open Source)**

* כל אחד יכול להוריד, להשתמש ולשפר.
* קהילת מפתחים תורמת ספריות (Packages) רבות.

**🖼️ ההבדל בין Front-End ל־Back-End**

| **תחום** | **Front-End** | **Back-End** |
| --- | --- | --- |
| מיקום | דפדפן (צד הלקוח) | שרת (Node.js לדוגמה) |
| שפות | HTML, CSS, JavaScript | JavaScript (Node.js), Python ועוד |
| תפקיד | עיצוב ואינטראקציה | לוגיקה, מסדי נתונים, APIs |

**🔌 מה זה API?**

* Application Programming Interface
* מתווך בין לקוח לשרת
* מאפשר לתוכנה אחת לתקשר עם אחרת

**🌍 Public API**

* API פתוח ונגיש לכולם
* אין צורך בהרשאות מיוחדות
* דוגמה: API של מזג אוויר

**🚀 Express.js**

* ספרייה לבניית שרתים ב־Node.js
* מקלה על הגדרת נתיבים, בקשות ותגובות

const express = require('express');

const app = express();

app.get('/', (req, res) => {

res.send('שלום מהשרת!');

});

app.listen(3000, () => {

console.log('השרת רץ על פורט 3000');

});

**🧠 Node.js – סיכום קצר**

* JavaScript בצד השרת
* מבוסס על מנוע V8 של כרום
* מתאים ל־APIs, מערכות זמן אמת, יישומים מבוזרים

**💬 פקודות בסיסיות ב־CLI**

| **פקודה** | **תיאור** |
| --- | --- |
| dir | מציג קבצים ותיקיות במיקום הנוכחי |
| cd | כניסה לתיקייה בשם מסוים |
| mkdir | יצירת תיקייה חדשה |
| type nul > name | יצירת קובץ חדש ריק בשם ובסיומת מסוימת |
| node index.js | הרצת קובץ JavaScript עם Node.js |

**🔧 התקנות ופקודות בסיסיות (שיעור 10)**

* **התקנת חבילה גלובלית:**
* npm install -g <package-name>
* **בדיקת חבילות גלובליות:**
* npm list -g --depth=0

**📂 עבודה עם מערכת קבצים (fs)**

* קיימות גרסאות סינכרוניות וא-סינכרוניות:

const fs = require('fs');

const data = fs.readFileSync('file.txt', 'utf8');

fs.readFile('file.txt', 'utf8', (err, data) => {

if (err) throw err;

console.log(data);

});

**🌐 HTTP ו־HTTPS**

* HTTP – תקשורת בין לקוח לשרת, לא מוצפן
* HTTPS – גרסה מאובטחת עם SSL/TLS

**🔗 URL**

* כולל: פרוטוקול, דומיין, נתיב (path)

**📡 Request/Response**

* הלקוח שולח בקשה – השרת מחזיר תגובה

**⚙️ פעולות CRUD**

* POST (יצירה), GET (קריאה), PUT (עדכון), DELETE (מחיקה)

**🔌 API ו־Endpoint**

* נקודת גישה בשרת לפעולה
* Header – מידע נלווה
* Body – נתונים שנשלחים

**🚀 Express – שרת פשוט**

const express = require('express');

const app = express();

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

app.listen(3000);

**🧱 Middleware**

* קוד שפועל על הבקשה לפני שהיא נשלחת ללקוח

**🖼 Static Files**

app.use(express.static('public'));

**📦 body-parser**

app.use(express.urlencoded({ extended: true }));

app.use(express.json());

**🔄 Fetch**

fetch('/api/user', {

method: 'POST',

headers: { 'Content-Type': 'application/json' },

body: JSON.stringify({ name: 'Avi', age: 30 })

});

**❓ Status Codes**

* 2XX – הצלחה
* 4XX – שגיאה מצד הלקוח
* 5XX – שגיאת שרת

**📝 תרגול לשיעור:**

1. מה זה Node.js ובמה הוא שונה מג׳אווהסקריפט רגילה?
2. מה ההבדל בין פקודת mkdir ל־cd?
3. כתוב קוד של שרת Express שמחזיר טקסט
4. מה זה API ציבורי ולמה הוא שימושי?
5. מה התפקיד של Header בבקשת fetch?
6. הסבר את ההבדל בין Front-End ל־Back-End
7. איך יוצרים קובץ חדש ב־CMD?
8. מה ההבדל בין HTTP ל־HTTPS?
9. מתי נשתמש ב־Middleware?
10. מה תפקיד body-parser ואיך משתמשים בו?