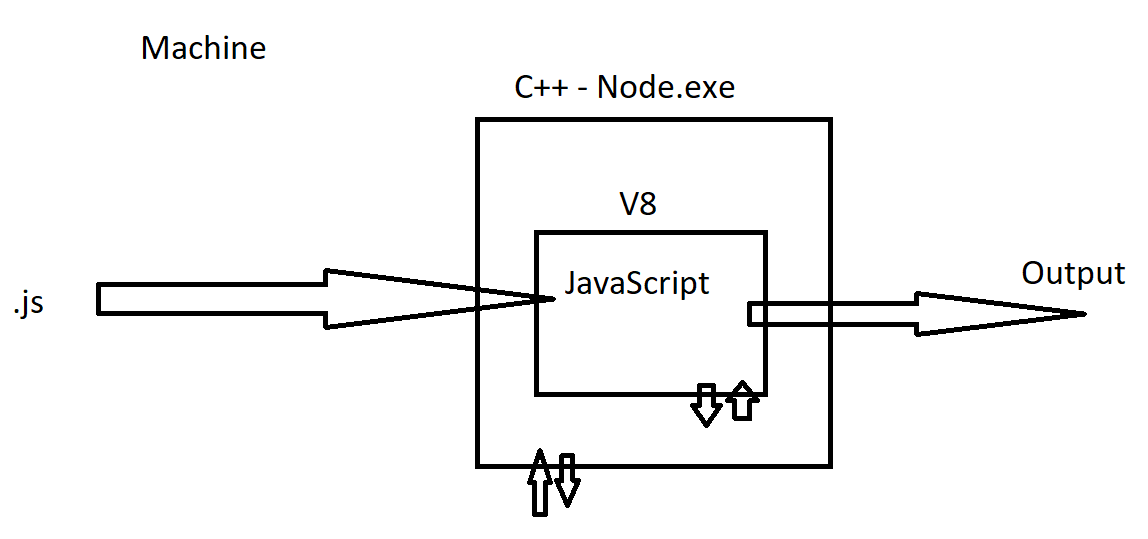
# **Node.js**

**Node.js**

סביבת הרצה של JavaScript מחוץ לדפדפן.



אין ב-Node.js:

* window
* DOM
* document
* תצוגה
* alert

יש בה בנוסף ל-JS במסגרת דפדפן:

* גישה ל-File System
* יצירה / קריאה של קבצים ותיקיות
* גישה למסדי נתונים
* משאבי מערכת ההפעלה

אתר רשמי: <https://nodejs.org/en>

**package.json**

זהו קובץ הגדרות ברמת פרויקט Node.js. מכיל שם, גרסה, dependencies ועוד.

**Package Manager**

מערכת הקיימת על שרת מרוחק והספקת ספריות עבור טכנולוגיית פיתוח כלשהי.

לדוגמה עבור C# קיים Package Manager בשם NuGet

לדוגמה עבור PHP קיים Package Manager בשם Composer

עבור Node.js קיים Package Manager בשם NPM – Node Package Manager.

**Package**

פרויקט Node.js שלם.

**global**

זהו האובייקט הגלובלי ב-Node.js (מקביל ל-window).

מכיל תשתית מסוימת עבור Node.js

**Module**

זהו קובץ js. אחד ויחיד.

כל קובץ כזה מכיל אוטומטית IIFE בתוכו בכדי לנתק את הקוד שלנו מהאובייקט הגלובלי – global

קיימים שלושה סוגי מודולים:

1. Custom Module – מודול שאנו בונים.
2. Built-In Module – מודול שקיים ב-Node.js כברירת מחדל. חלק מתשתית Node.js.
3. NPM Module – מודול הקיים ב-Package Manager שאנו עובדים איתו.

התקנה: npm install module-name

או בקיצור: npm i module-name

הסרה: npm uninstall module-name

קיים גם סוג של Module שאנו יכולים להתקין ברמת כל המחשב.

npm install --global module-name

או בקיצור: npm i -g module-name

זאת אך ורק עבור ספריות המכילות פקודות CLI – Command-Line Interface

**nodemon**

זו ספרייה המאפשרת להריץ פרויקט Node ע"י מנגנון Hot-Reloading

קיצור של Node Monitor

זו ספרייה המכילה פקודת CLI להרצת הפרויקט ולכן היא מותקנת ע"י npm i -g nodemon

**Promise**

זהו אובייקט Native המאפשר להריץ קוד אסינכרוני ויודע לדווח לנו ברגע שהסתיים – הצלחה או כשלון.

כמה שנים לאחר המצאת האובייקט – התווספו שתי מילים שמורות ל-JavaScript: async ו-await.

מילים אלו פועלות אך ורק על Promise.

async מוגדרת לפני הפונקציה שאנו בונים בכדי לציין למערכת שאנו כותבים פונקציה המכילה קוד אסינכרוני.

await נכתבת לפני אובייקט ה-Promise. היא מבצעת מספר דברים:

1. היא ממתינה לסיום פעילות ה-Promise.
2. ההמתנה לא תוקעת את כל המערכת אלא רק את הפונקציה הנוכחית שלנו ומשחררת את כל JavaScript להמשך פעילות.
3. מחזירה את התוצאה שאובייקט ה-Promise מדווח בעת הצלחה.

**Observable**

אובייקט המיועד להריץ קוד אסינכרוני שמדווח רצף של נתונים לאורך זמן.

ניתן להירשם אליו ולקבל את הדיווחים ממספר מקומות בקוד.

יכול לדווח שגיאה אחת (ואז הוא נסגר).

צריך לדווח אירוע נוסף של complete ברגע שרצף המידע הסתיים.

**To Promisify**

הפיכת קוד ישן, שעובד ע"י successCallback/errorCallback לכזה שעובד ע"י Promise

בכדי שנוכל לקרוא לקוד זה ע"י async-await.

**TypeScript**

שפת תכנות שנכתבה ע"י מייקרוסופט ב-2013.

מכילה קומפיילר שהופך את הקוד ל-JavaScript.

יתרונות:

1. TypeScript הינה Strongly Typed כאשר JavaScript אינה.
2. יש לה קומפיילר המבצע שני דברים: 1. בודק שגיאות ומתריע. 2 הופך את הקוד ל-Native JavaScript (נקרא Transpiler).
3. בגלל שהיא Strongly Typed קיים IntelliSense כמו שצריך.
4. יש לה תחביר OOP מלא, בדומה לשפות מתקדמות כמו #C או Java כאשר ה-OOP של JavaScript לוקה בחסר!
5. ועוד...

התקנת הקומפיילר של TypeScript:

npm i -g typescript

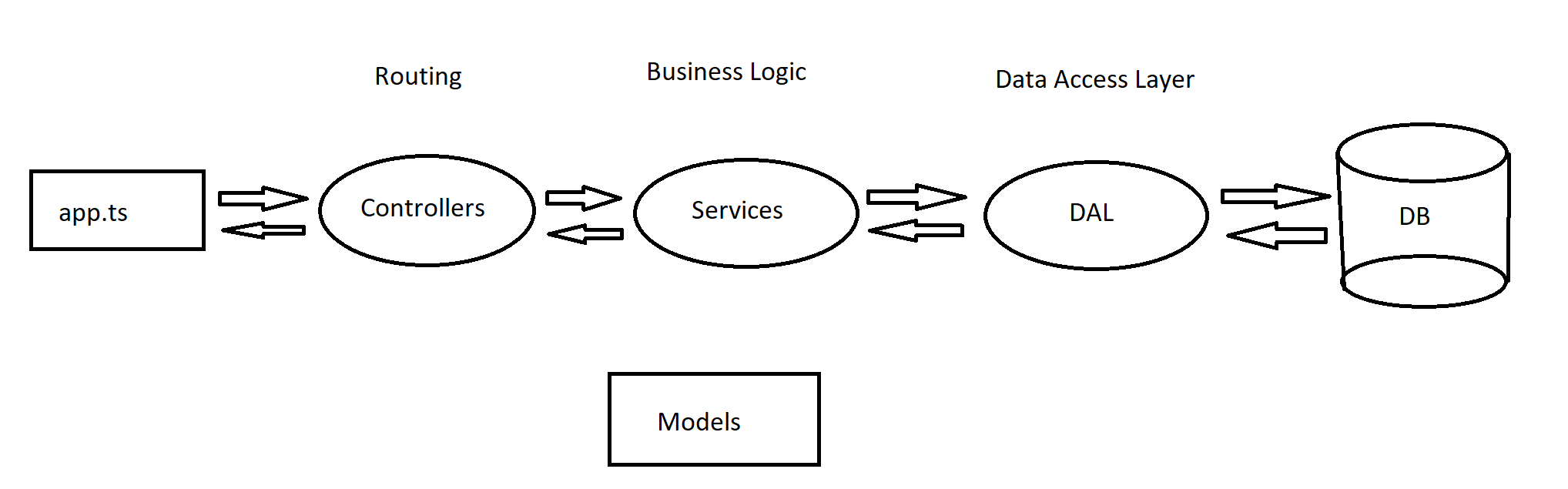
או לחילופין – התקנת ts-node המאפשר להריץ קוד Node.js שנכתב ב-TypeScript:

npm i -g ts-node

**Express**

ספריית Node.js לבניית REST API.

**Layered REST API**



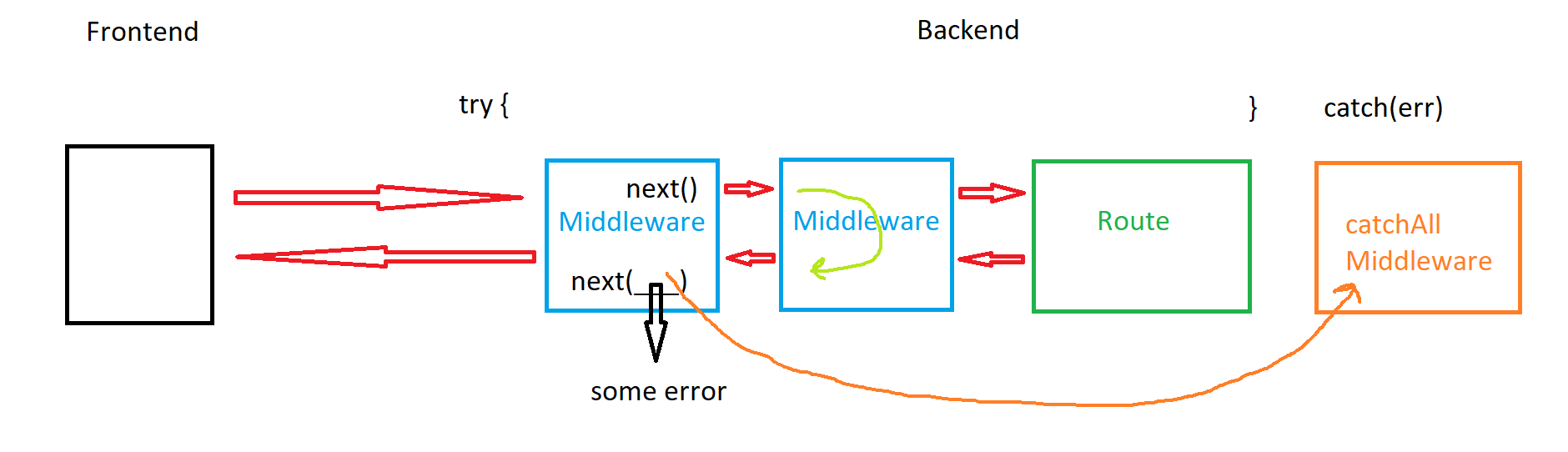
**Middleware**

זו פונקציה המתבצעת בצורה אוטומטית בין ה-Request לבין ה-Response.

Middleware יכול להתבצע אוטומטית עבור כל ה-Requests, עבור חלק מהם או עבור Request ספציפי.

Middleware יכול לבצע פעולה ולהמשיך את ה-Request הלאה ליעד שלו.

Middleware יכול להפסיק את ה-Request ומידית להחזיר Response כאשר במקרה כזה ה-Request לא יגיע ליעד שלו.



DB

ישנם שני סוגים של מסדי נתונים:

1. מסד נתונים רלציוני – Relational Database.

זהו מסד נתונים המכיל טבלאות וקישורים בין הטבלאות.

שפת SQL פועלת על מסדי נתונים אלו. לכן יותר קל לבצע שאילתות מורכבות.

דוגמאות: MySQL, Postgress, MSSQL – Sql Server, Oracle Server...

1. NoSQL

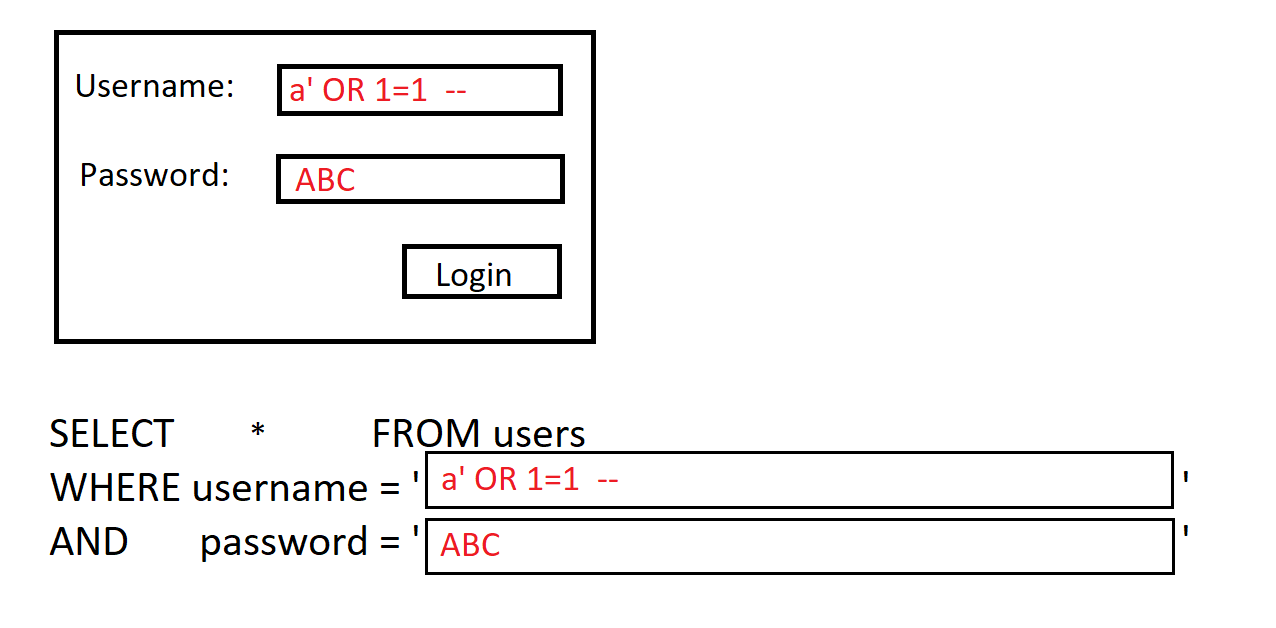
זהו מסד נתונים ללא טבלאות. המידע נמצא בו בפורמט אחר. לדוגמה JSON.

דוגמאות: MongoDB, Elastic Search, Cassandra, ...

אלו מסדי נתונים מהירים יותר המתאימים גם ל-BigData.

**SQL Injection**

התקפה נפוצה הנגרמת ע"י "הזרקת" חלק של שאילתת SQL לתוך תיבת טקסט או ערוץ קלט אחר, המשתלבת בשאילתה המקורית ומשנה את המשמעות שלה לצורך פריצה, גניבת מידע, מחיקת מידע, שיבוש טבלאות וכדומה.



**טיפול בקבצים**

ספרייה לטיפול בקבצים הנשלחים מצד הלקוח:

npm i express-fileupload

npm i @types/express-fileupload -D

ספרייה ליצירת UUID:

npm i uuid

npm i @types/uuid -D