# **Node.js**

**Node.js**

סביבת הרצה של JavaScript מחוץ לדפדפן.

זה מאפשר ל-JavaScript לרוץ על מחשב שרת.

אפשר להריץ TypeScript על Node.js.

אין ב-Node.js את אובייקט ה-window, אין את אובייקט ה-document, אין alert, אין כלל דפדפן.

**Package**

פרויקט שלם ב-Node.js. בנוי ממספר קבצי JavaScript (או TypeScript).

**Module**

זהו קובץ אחד של JavaScript (או TypeScript).

Node.js אוטומטית עוטפת את הקוד שבכל קובץ ב-IIFE (Immediately Invoked Function Expression).

קיימים שלושה סוגים של Modules:

1. Custom Module – זהו קובץ js. שאנו בונים בפרויקט שלנו. שם ה-Module זה שם הקובץ (ללא הסיומת).
2. NPM Module – התקנת package (המכיל מודולים) משרתי NPM. ספריות אלו מותקנות בתוך node\_modules.
3. Built-In Module – מודול קיים ב-Node.js, לא דורש שום התקנה.

**package.json**

זהו קובץ החייב להיות בפרויקט Node.js המכיל הגדרות ברמת הפרויקט.

פקודה ליצירת קובץ כזה: npm init -y

**Nodemon**

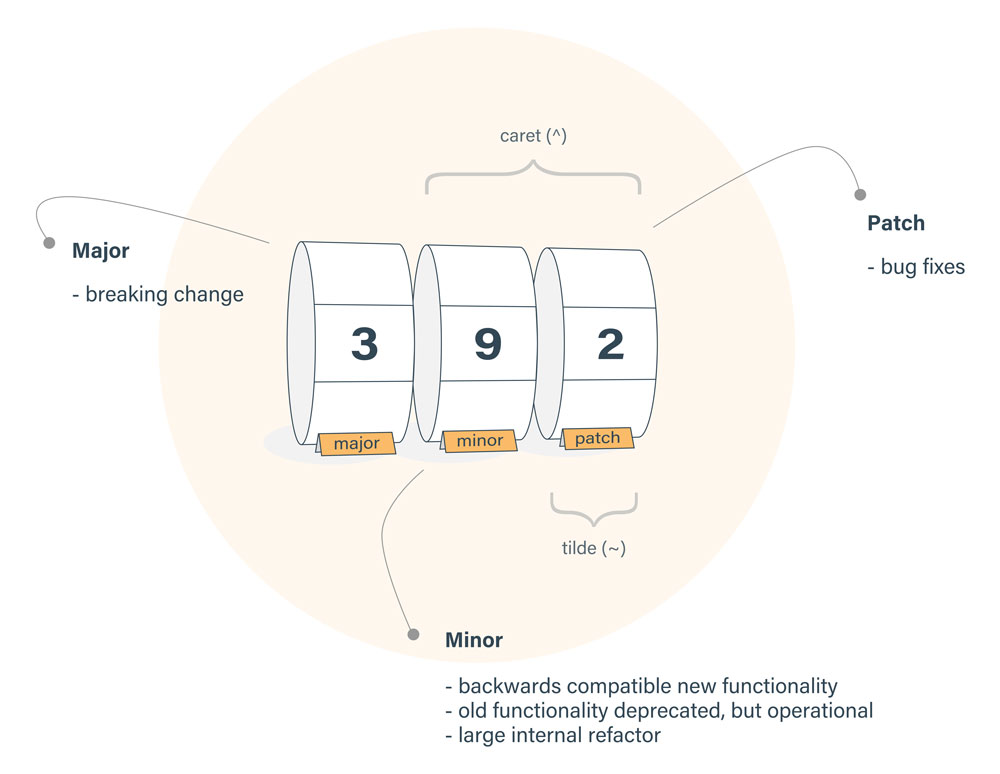
ספרייה המותקנת גלובלית והמאפשרת להריץ פרויקט Node.js ע"י Hot-Reloading.

(קיצור של Node Monitor).

התקנה: npm i -g nodemon

רצת פרויקט ע"י Nodemon: nodemon app.js

Versions



שחזור פרויקטים (npm i)

התקנת הגרסה המדויקת איתה אנו עובדים:

"cute-kittens": "2.3.7"

התקנת הגרסה המכילה את ה-Bug-Fix החדשים ביותר:

"cute-kittens": "~2.3.7"

התקנת הגרסה המכילה את ה-Minor Release החדש ביותר:

"cute-kittens": "^2.3.7"

התקנת הגרסה החדשה ביותר (לא מומלץ כי בוודאות ישבור את הקוד מתישהו):

"cute-kittens": "\*"

package-lock.json מכיל את המספרים המדויקים של כלל ההתקנות.

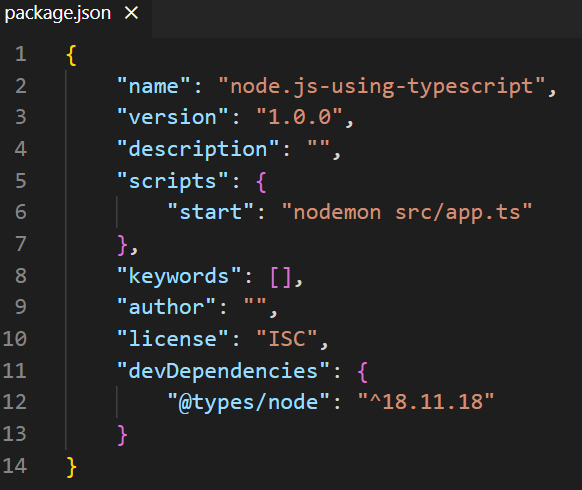
אם נשחזר את הפרויקט ע"י הפקודה npm ci – נקבל את הגרסאות המדויקות שקיימות ב-package-lock ולא את כלל החידושים.

כלי לערפול קוד JavaScript:

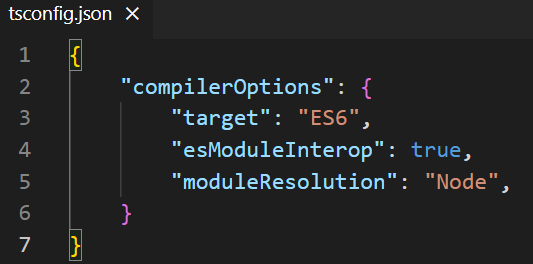
npm i -g javascript-obfuscator

**יצירת פרויקט TypeScript ב-Node.js**

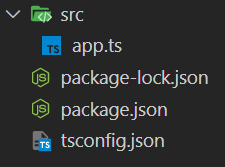
1. התקנה חד פעמית ברמת המחשב המאפשרת להריץ TypeScript: npm i -g ts-node
2. התקנת ספריית Types עבור הטיפוסים הבסיסיים של Node.js ב-TypeScript: npm i @types/node -D
3. קובץ ה-package.json:



1. יצירת קובץ tsconfig.json:



1. הקבצים והתיקיות:



**REST API**

REST – Representational State Transfer

שירות הקיים על שרת כלשהו שמנגיש מידע על גבי תקשורת HTTP (רשת האינטרנט).

לרוב המידע קיים ב-Database (מסד נתונים)

לקוח מבצע HTTP Request לשירות ה-REST API. כאן הוא מגיע לפונקציה ספציפית שמחזירה לו את המידע ב-HTTP Response.

שירות שבנוי בדיוק לפי ארכיטקטורת REST API נקרא RESTful Service.

**Express**

זו ספרייה המאפשרת לבנות REST API בפרויקט Node.js.

היא מאפשרת בצורה פשוטה להאזין על Method מסוים + Route מסוים ולהחזיר מידע כאשר גולשים ל-Method + Route אלו.

התקנות:

npm i express

npm i @types/express -D

**Postman**

כלי פופולרי לצורך בדיקת REST API שאנו בונים.

הוא מדמה Frontend ומאפשר לשלוח כל דבר שאנו רוצים ב-Request ולהתבונן במידע המוחזר ב-Response.

<https://www.postman.com/downloads>

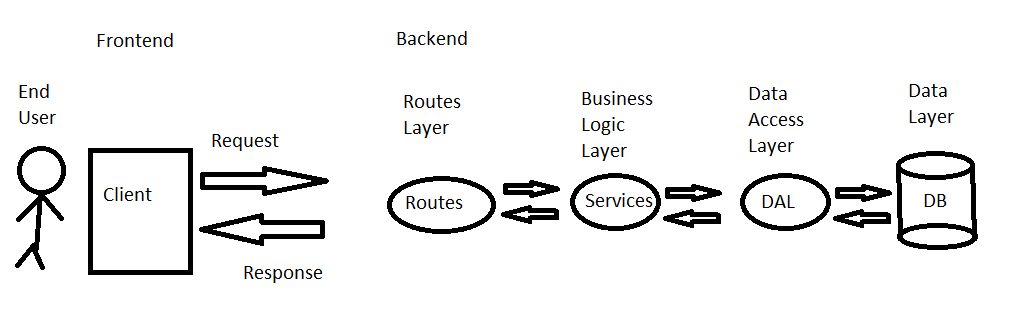
**עבודה במודל שכבות**

בניית REST API בחלוקת הפרויקט לשכבות.

שכבה = Layer

שכבה הינה אזור בקוד שאחראי על פעילות מסוימת במערכת.

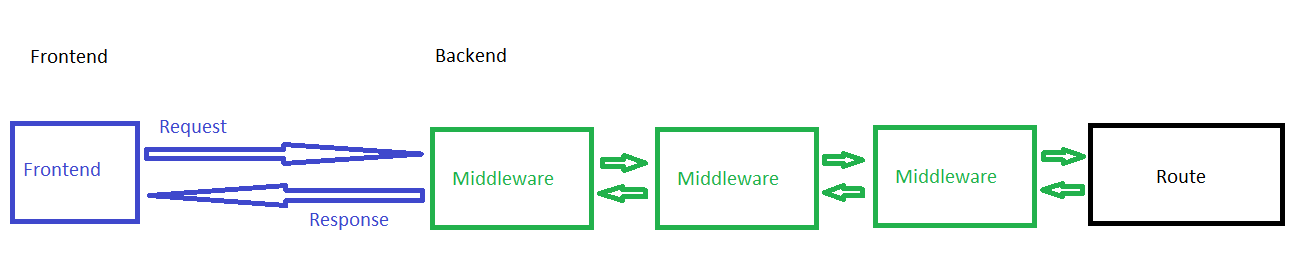
כך המערכת (ה-REST API) מחולקת למספר קבצים לפי תחומי האחריות.



**Middleware**

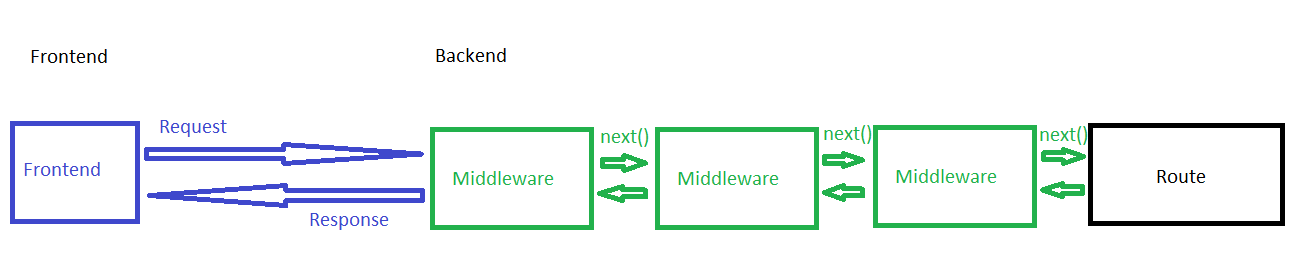
זו פונקציה המתבצעת אוטומטית בין ה-Request לבין ה-Response.

כל דבר שאנו רוצים לבצע אוטומטית ניתן לעשות ב-Middleware כמו כתיבה ל-Logs, מניעת מידע זדוני, בדיקות Tokens...

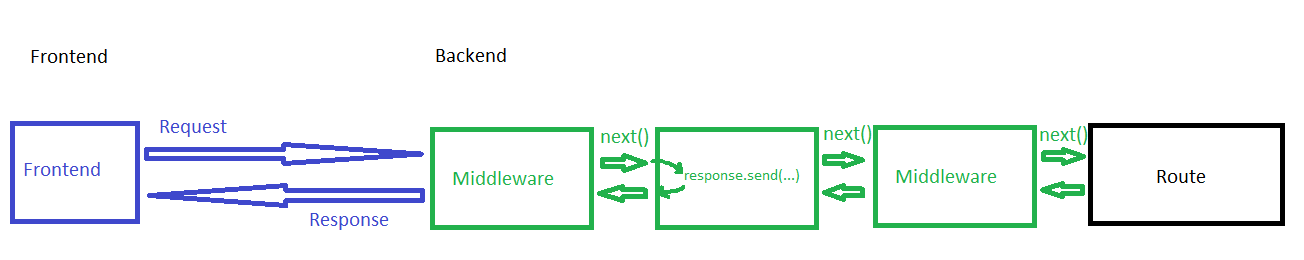


אפשר לרשום Middleware ברמת כל האתר או ברמת Route ספציפי.

Middleware יכול לבצע פעילות כלשהי ואז להמשיך את הזרימה ל-Middleware הבא או ל-Route (אם הוא קיים בדיוק לפני ה-Route). זאת ע"י קריאה לפונקציה next():



Middleware יכול לעצור את הזרימה ובמקום להגיע ל-Middleware הבא או ל-Route – להחזיר מידית Repsonse.

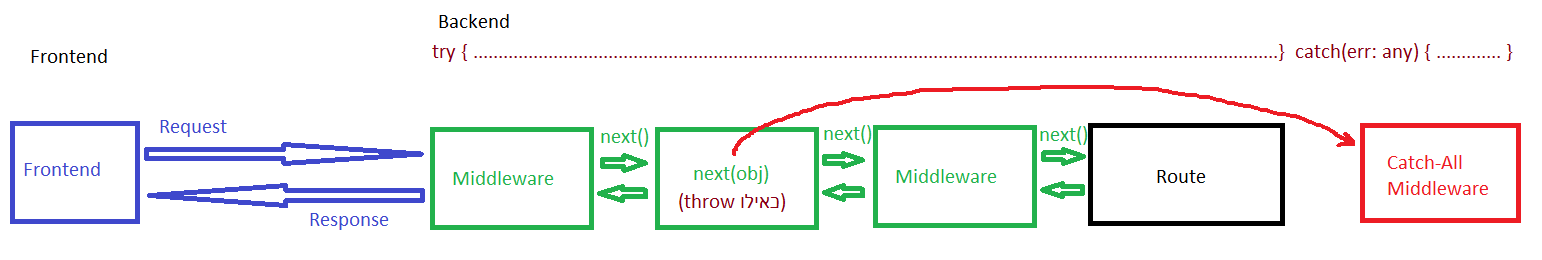


קיים Middleware מיוחד המקבל ארבעה ארגומנטים:

1. אובייקט שגיאה
2. אובייקט request
3. אובייקט response
4. פונקציית next

רושמים את ה-Middleware הזה אחרי כלל ה-Routes.

אם אנו שולחים אובייקט לפונקציית ה-next ב-Middleware מסוים – המערכת תדלג מעל כל היתר ותגיע ישירות ל-Middleware המיוחד הזה.



טיפול בשגיאות

ישנן מספר סוגי שגיאות.

במקרה של שגיאה יש להחזיר ל-Front את הסטטוס המתאים לה:

קריסה בשרת: 500

נתיב שלא נמצא: 404

Resource (לפי id) שלא נמצא: 404

שגיאת וולידציה: 400

שגיאת אימות: 401

שגיאת הרשאות: 403

בכל מקרה של שגיאה עלינו לנתב ל-Catch All לצורך טיפול אחד ויחיד בכל השגיאות.

שגיאות וולידציה בודקים ע"י ספרייה ייעודית לכך בשם Joi

התקנה:

npm i joi

**UUID**

Universal Unique IDentifier

זהו טקסט ייחודי שאלגוריתם מייצר כך שהטקסט לא יחזור על עצמו לעולם.

משתמשים בזה לצורך יצירת קודים ארוכים, שמות קבצים ייחודיים ועוד.

מונח דומה הינו GUID – Global Unique IDentifier – אותו הדבר, אך המונח בשימוש ע"י מייקרוסופט.

ספרייה ליצירת UUID:

npm i uuid

npm i @types/uuid -D

ספרייה לקבלת תמונה שה-Front שלח:

npm i express-fileupload

npm i @types/express-fileupload -D