



Node.js® is a JavaScript runtime built on [Chrome's V8 JavaScript engine](#).

Created by: David Yakin 2022

תוכן עניינים

4	<u>Introduction to node.js</u>	▶
8	<u>Install</u>	▶
11	<u>Entry point</u>	▶
12	<u>Global object</u>	▶
17	<u>Module system</u>	▶
18	<u>Require</u>	▶
24	<u>Core modules</u>	▶
26	<u>OS</u>	▶
29	<u>FS</u>	▶
39	<u>Path</u>	▶
43	<u>External Libraries</u>	▶
44	<u>Npm</u>	▶
50	<u>npm install</u>	▶
52	<u>Global</u>	▶
54	<u>Dependencies</u>	▶
58	<u>devDependencies</u>	▶
60	<u>package-lock.json</u>	▶

61	chalk	▶
70	External Libraries	▶
71	LODASH	▶
75	Hexagonal architecture	▶
88	Joi	▶
96	Normalize Objects	▶
100	Mongoose	▶
109	Mongoose model	▶
126	Mongoose Queries	▶
144	Mongoose Aggregation Operations	▶
156	Bcryptjs	▶
161	Environments	▶
162	Config	▶
168	Authentication & authorization	▶
169	Json Web Token	▶
183	Initial Data	▶



definition

Node.js is an asynchronous event-driven JavaScript development platform and runtime environment that runs on Chrome's V8 engine that executing JavaScript code server-side.



לינק להרצאה ראשונה של ryan dahl של node.js

<https://www.youtube.com/watch?v=ztspvPYyblY>

היתרונות של Node.js

JavaScript בצד שרת

גישה גם לחומרת המחשב באמצעות פונקציות אסינכרוניות javascript

Common js module system

Input Output (i.o) - single thread - event loop

אופטימיזציה של המערכת כך שה - event loop לא ייתקע (call backs)

"JavaScript every where" paradigm

Global object & variables - global process ...

שימוש כמה שניתן במילים שמורות של javascript בצד לקוח

תמיכה בספריות חיצוניות (npm yarn וכדומה)

קישור לאתר של node.js

<https://nodejs.org/en/>

התקנת node.js

ישנן שתי גרסאות אותן ניתן להוריד: ►

► LTS – הגרסה הבטוחה

► Current – הגרסה העדכנית ביותר



Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine.

Download for Windows (x64)

16.17.0 LTS

Recommended For Most Users

18.9.0 Current

Latest Features

[Other Downloads](#) | [Changelog](#) | [API Docs](#)

[Other Downloads](#) | [Changelog](#) | [API Docs](#)

Or have a look at the Long Term Support (LTS) schedule

* נלחץ על הכפתור ונוריד את גרסת LTS

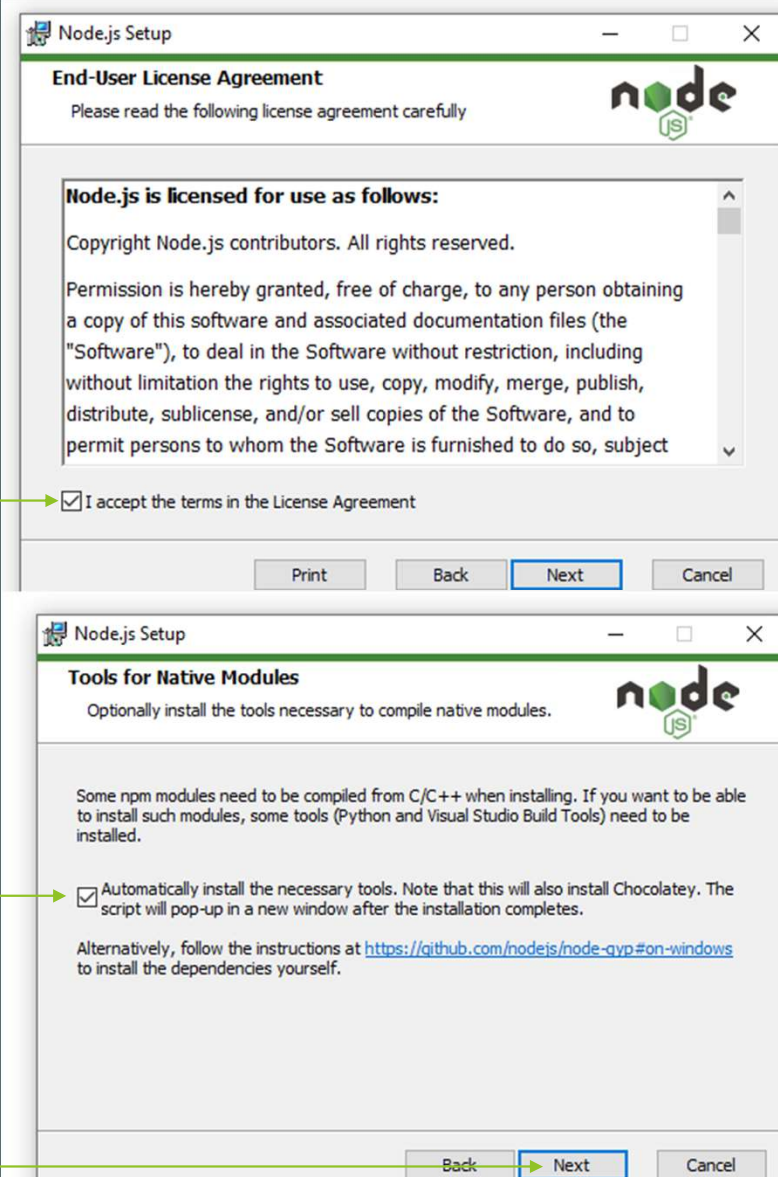
<https://nodejs.org/en/>



Installation wizard

באשף ההתקנה של node.js תצטרכו לסמן בשני מקומות בתיבות הסימון וללחוץ על כפתור next בכל שאר החלונות הנפתחים אפשר ללחוץ ישירות על כפתור ה - next

* בעזרת הרצת הפקודה `node -v` ב - cmd ניתן לראות את הגרסת התוכנה שהותקנה על המחשב



javascript בצד שרת

עכשיו שתוכנת node.js מותקנת לנו על המחשב אנו יכולים ולהבין טוב יותר מה זה אומר הרצת שפת javascript בצד שרת.

בדוגמה הראשונה אני מנסה להריץ ב – terminal קוד javascript וכשאני לוחץ על מקש Enter אני מקבל את הודעת השגיאה הבאה

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER
C:\Users\DELL\Desktop\HackerU\lecturer-work\Lessons\NODE.JS\NODE.JS-SANDBOX>
const obj = {name: "david"};
'const' is not recognized as an internal or external command,
operable program or batch file.
C:\Users\DELL\Desktop\HackerU\lecturer-work\Lessons\NODE.JS\NODE.JS-SANDBOX>
```

בדוגמה השנייה אני קודם כל מריץ את תוכנת node.js באמצעות הפקודה node

וכשאני כותב את אותו הקוד עכשיו הטרמינל מביין את משמעות הקוד ממש כמו הדפדפן

```
C:\Users\DELL\Desktop\HackerU\lecturer-work\Lessons\NODE.JS\NODE.JS-SANDBOX>
node
Welcome to Node.js v16.15.0.
Type ".help" for more information.
> const obj = {name: "david"};
undefined
> console.log(obj.name)
david
undefined
>
```

Entry point

קובץ ראשי אליו כל קבצי אפליקציה מחוברים
וכשמפעילים אותו הוא מריץ את הקוד
באפליקציה

בדוגמה שלהלן:

יצרתי קובץ בשם main.js

בתוכו הפעלתי את מטודת console.log עם
הכיתוב הבא

בעזרת הפקודה בטרמינל "filename" node
אני מפעיל את הקובץ Node יבצע את הקוד
ויציג לנו את הכיתוב שבפונקציה.

```
JS main.js X
JS main.js
1 console.log("In our entry point - the main.js file!");
2

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

C:\Users\DELL\Desktop\HackerU\lecturer-work\Lessons\NODE.JS\NODE.JS-SANDBOX>
node main
In our entry point - the main.js file!
```

! אין צורך לכתוב בטרמינל את הסיומת js לאחר הפקודה node

Global object

אובייקט גלובלי שניתן להגיע אליו מכל מודול באפליקציה ואפילו מחוצה לה.

וכמובן שאם ניתן להגיע אליו ניתן גם לעשות השמה למשתנים, פונקציות ומחלקות חדשות בתוכו.

בדוגמה שלהלן:

אני מוסיף לאובייקט global מפתח בשם user ומשווה את הערך שלו לאובייקט עם מפתחות וערכים

מרגע זה אני יכול לגשת אל אותו מפתח user וערכיו מכל מקום באפליקציה ו - node.js תזהה אותו כמפתח כמו שאר המפתחות באובייקט ה - global

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
C:\Users\DELL\Desktop\HackerU\lecturer-work\Lessons\NODE.JS\NODE.JS-SANDBOX>node
Welcome to Node.js v16.15.0.
Type ".help" for more information.
> global.user={name: 'david', age: 47}
{ name: 'david', age: 47 }
> global.user.age
47
> 
```



גישה למפתחות באובייקט ה - Global

בדומה לאובייקט ה - window בדפדפן גם במקרה של global ניתן לגשת ישירות את המשתנים והמטודות שבתוכו.

בדוגמה שלהלן:

אני עושה מוסיף את המפתח user לאובייקט global ומשווה את הערך שלו לאובייקט

אני מפעיל את פונקציית console.log ישירות עם שם המפתח מבלי לרשום לפניו global ומקבל את התשובה כאילו כן רשמתי.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  JUPYTER

C:\Users\DELL\Desktop\HackerU\lecturer-work\Lessons\NODE.JS\NODE.JS-SANDBOX>node
Welcome to Node.js v16.15.0.
Type ".help" for more information.
> global.user = {name: "david", age: 47}
{ name: 'david', age: 47 }
> console.log(user.age)
47
undefined
> 
```

! לינק לרשימה המלאה של המטודות, המשתנים והמחלקות באובייקט global
<https://nodejs.org/dist/latest-v16.x/docs/api/globals.html>

Execution global functions

באותה צורה יש לי גישה ישירה למטודות שנמצאות בתוך אובייקט ה - global

בדוגמה שלהלן אני מפעיל את הפונקציה `setTimeout` global שנמצאת בתוך אובייקט ה - global

כמו בדפדפן גם בטרמינל כל הפעלת שורה תיתן לי חיווי למה שהפונקציה/ משתנה מחזיר מיד

וכעבור שלוש שניות יודפס לי בקונסול הקוד הבא

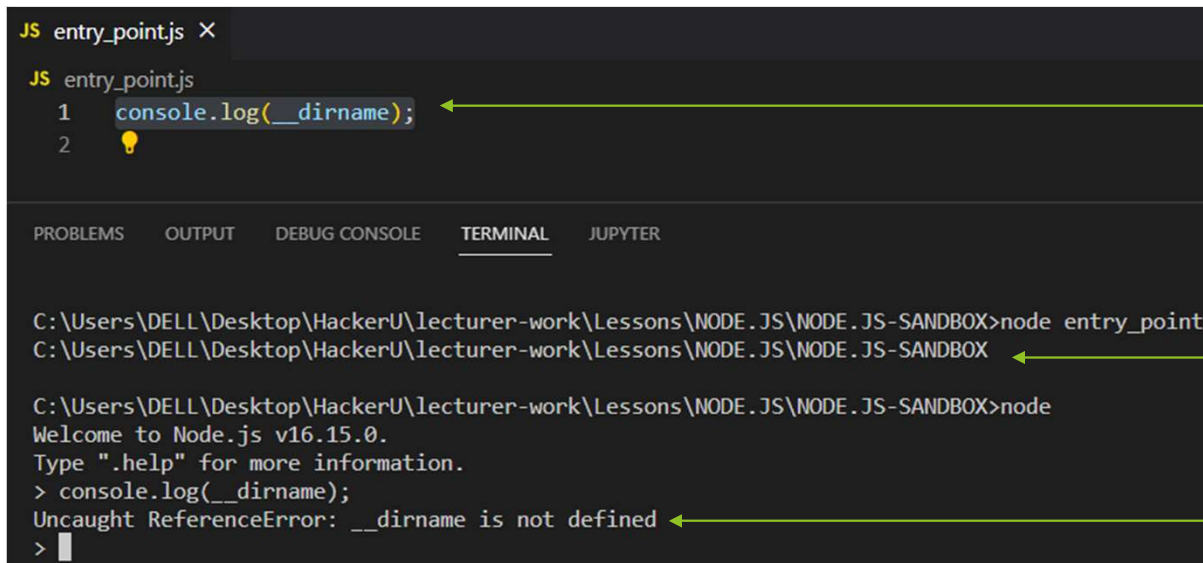
```
> setTimeout(()=>console.log('setTimeout in the global object!'),3000);
Timeout {
  _idleTimeout: 3000,
  _idlePrev: [TimersList],
  _idleNext: [TimersList],
  _idleStart: 1756432,
  _onTimeout: [Function (anonymous)],
  _timerArgs: undefined,
  _repeat: null,
  _destroyed: false,
  [Symbol(refed)]: true,
  [Symbol(kHasPrimitive)]: false,
  [Symbol(asyncId)]: 1325,
  [Symbol(triggerId)]: 5
}
> setTimeout in the global object!
```

Global objects in the module scope

ישנם משתנים שמתנהגים כגלובליים בתוך האפליקציה אולם הם חיים רק בסקופ של המודולים ולא בטרמינל

בדוגמה שלהלן ניתן לראות שכאשר אני משתמש באובייקט ה"גלובלי" `__dirname` בתוך ה- `scope` של המודול, `node.js` מזהה את האובייקט ומחזיר את הכתובת הנוכחית של המודול המבוקש

אולם אם אפעיל את התוכנה `node.js` בעזרת הפקודה `node` ונסה להשתמש באותה פונקציה. `Node` יזרוק לי שגיאה ש- `__dirname` לא מוגדר זאת בגלל שלא ניסיתי לגשת אליו בתוך הסקופ של המודול.



The screenshot shows a code editor with a file named `entry_point.js` containing two lines of JavaScript code:

```
1 console.log(__dirname);  
2
```

Below the editor is a terminal window. The first command executed is `node entry_point`, which returns the file path: `C:\Users\DELL\Desktop\HackerU\lecturer-work\Lessons\NODE.JS\NODE.JS-SANDBOX`. The second command is `node`, which starts the Node.js REPL. In the REPL, the command `console.log(__dirname);` is entered, resulting in an error: `Uncaught ReferenceError: __dirname is not defined`. Green arrows point from the text on the right to the code in the editor and the error message in the terminal.

! לינק לרשימה המלאה של האובייקטים שהם גלובליים רק בסקופ של המודול:

<https://nodejs.org/dist/latest-v16.x/docs/api/globals.html#global-objects>

Cmd commands

נא לא לבלבל בין משתנים, פונקציות ומחלקות של node.js לפקודות של ה-cmd

נוכל בקלות להבדיל בין השניים העזרת הכלל הבא:

אם הטרימינל מזהה את הפקודה, היא פקודה של ה-cmd.

(במקרה הזה הפקודה יוצרת תיקייה עם השם שניתן לה)

אם הטרימינל מזהה את המשתנה / פונקציה רק לאחר כתיבת שורת הקוד node אזי מדובר במשתנה, פונקציה או מחלקה של node.js

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

C:\Users\DELL\Desktop\HackerU\lecturer-work\Lessons\NODE.JS\NODE.JS-SANDBOX>
mkdir app.js

C:\Users\DELL\Desktop\HackerU\lecturer-work\Lessons\NODE.JS\NODE.JS-SANDBOX>
```

```
✓ NODE.JS-SANDBOX
> app.js
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

C:\Users\DELL\Desktop\HackerU\lecturer-work\Lessons\NODE.JS\NODE.JS-SANDBOX>
console.log("cmd doesn't recognize node.js recognizes me!");
'console.log' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\DELL\Desktop\HackerU\lecturer-work\Lessons\NODE.JS\NODE.JS-SANDBOX>
node
Welcome to Node.js v16.15.0.
Type ".help" for more information.
> console.log("node.js recognizes me!");
node.js recognizes me!
undefined
>
```




▶ Module system

רכישת קוד ממודולים שונים באפליקציה



Require method

Used to import modules, JSON, and local files.



רכישת מודול

פונקציית require משמשת אותנו כדי לרכוש קוד ממודול אחר.

בדוגמה שלהלן:

בקובץ exports.js - אני מפעיל את מטודת console.log כשאני מעביר לה בפרמטר טקסט מסוים

בקובץ main.js - אני רוכש את הקוד והוא יופעל מיד עם הפעלת המודול הזה שימש אותנו כ - entry point לאפליקציה מכיוון שהוא כתוב בשורה הראשונה.

ניתן לראות שבהפעלת הקוד מהקובץ main מבוצע הקוד שרכשתי מהקובץ exports.js

```
JS exports.js ×
EXPORTS_MODULE > JS exports.js
1 console.log("in exports!");
2

JS main.js ×
EXPORTS_MODULE > JS main.js
1 require("./exports");
2

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

C:\Users\DELL\Desktop\HackerU\lecturer-work\Lessons\
NODE.JS\NODE.JS-SANDBOX\EXPORTS_MODULE>node main
in exports!
```

Module.exports

במקרה בו אנו נרצה לייבא קוד ממודול אחר ולהפעיל אותו במודול שמייבא אותו ניתן להשתמש במס' דרכים והראשונה היא באמצעות אובייקט module.exports

בדוגמה שלהלן:

בקובץ module.exports.js

אני יוצר את מטודת fn

אני מייצא אותה באמצעות module.exports

בקובץ main.js -

אני רוכש את הפונקציה ושם אותה בתוך משתנה בשם fn

אני מפעיל את המטודה מתוך המודול הזה

ניתן לראות שבהפעלת הקוד מהקובץ main פונקציית fn מופעלת ומביאה את תוצריה למרות שכתובת הפונקציה נעשתה במודול אחד והפעלתו קורית במודול אחר.

```
JS module.exports.js X
EXPORTS_MODULE > module.exports > JS module.exports.js > ...
1 const fn = () => console.log("From module.exports.js file!");
2
3 module.exports = fn;

JS main.js X
EXPORTS_MODULE > module.exports > JS main.js > ...
1 const fn = require("./module.exports");
2 fn();
3

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER
C:\Users\DELL\Desktop\HackerU\lecturer-work\Lessons\NODE.JS\NODE.JS-S
ANDBOX\EXPORTS_MODULE\module.exports>node main
From module.exports.js file!
```

! למרות שאין חובה כזאת כדאי לקרוא לשם המיכל (const/ let) בשם הפונקציה/ האובייקט/ המשתנה שאנו מייבאים

Exports.keys

דרך נוספת לייצא קוד ממודול הוא על ידי השמה של מפתחות לאובייקט ה-
exports

בדוגמה שלהלן:

בקובץ exports.key.js

אני את הקבוע first ומשווה אותו לאובייקט

אני יוצר קבוע בשם second ומשווה אותו למערך של מספרים

אני מייצא את שני הקבועים באמצעות השמה שלהם כמפתחות באובייקט ה- exports

בקובץ main.js -

הדרך לרכוש את המפתחות הללו היא באמצעות object destructurer

לאחר שחילצתי את המפתחות הם זמינים אלי במודול ואני יכול להשתמש בהם בדרכים שונות כמו להדפיס ערכים מסוימים מתוכם.

בהפעלת הקוד מהקובץ main הערכים מודפסים בטרמינל הערכים שביקשתי להדפיס

```
JS exports.key.js X
EXPORTS_MODULE > exports.key > JS exports.key.js > ...
1  const first = { name: "david", age: 43 };
2  const second = [1, 2, 3];
3
4  exports.first = first;
5  exports.second = second;

JS main.js X
EXPORTS_MODULE > exports.key > JS main.js > ...
1  const { first, second } = require("./exports.key");
2
3  console.log(first.name);
4  console.log(second[1]);
5

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER
C:\Users\DELL\Desktop\HackerU\lecturer-work\Lessons\NODE.JS\NODE.JS-S
ANDBOX\EXPORTS_MODULE\exports.key>node main
david
2
```

! הייתרון בשיטה זאת הוא שאני יכול לייצא מהמודול רק את הקוד שאני מעוניין לשתף עם המודולים האחרים

Exports.keys

אני יכול לייבא את כל מודול ה- exports מאותו קובץ ולראות את ההשמה בפועל של המפתחות והערכים שערכתי בקובץ הקודם.

בדוגמה שלהלן:

קובץ exports.key.js - נשאר אותו דבר

בקובץ main.js -

אני רוכש את המודול exports.key אל תוך קבוע בשם obj

אני מדפיס את האובייקט obj

בהפעלת הקוד מהקובץ main ניתן לראות שמודפס לי אובייקט ה- exports המפתחות והערכים שעשיתי להם השמה במודול exports.key.js

```
JS exports.key.js ×
EXPORTS_MODULE > exports.key > JS exports.key.js > ...
1  const first = { name: "david", age: 43 };
2  const second = [1, 2, 3];
3
4  exports.first = first;
5  exports.second = second;

JS main.js ×
EXPORTS_MODULE > exports.key > JS main.js > ...
6  const obj = require("./exports.key");
7  console.log(obj);
8

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  JUPYTER

C:\Users\DELL\Desktop\HackerU\lecturer-work\Lessons\NODE.JS\NODE.JS-S
ANDBOX\EXPORTS_MODULE\exports.key>node main
{ first: { name: 'david', age: 43 }, second: [ 1, 2, 3 ] }
```

משימת Module

- ▶ יצא ממודול אחד את שמך הפרטי באמצעות `module.exports`
- ▶ יצא ממודול שני את שם משפחתך באמצעות השמת מפתחות לאובייקט `exports`
- ▶ ייבא את השם הפרטי ואת שם המשפחה שלך מהמודולים השונים והדפס אותם בקונסול.





▶ Core modules

ספריות שמגיעות עם התקנת node.js



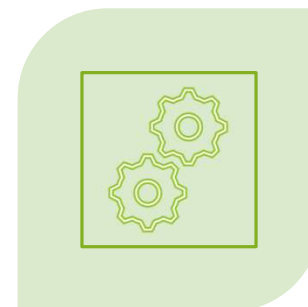
על מנת להבין את הכוח הטמון ב – node.js נכיר שלוש ספריות שנמצאות ב – core modules של התוכנה



PATH



FS
(FILE SYSTEM)



OS
(OPERATING SYSTEM)



ספרייה שמאפשרת לנו גישה
לנתוני המחשב ומערכת
ההפעלה.



OS

בדוגמה שלהלן:

ניצור קבוע בשם `OperatingSystem` שערכו יהיה הערך שיחזור מהפעלת מטודת `require` כאשר אני מעביר לה בפרמטר את שם הספרייה `os`

נעשה שימוש בשני מהמטודות מתוך ספרייה זאת:

`totalmem` – כדי לקבל את גודל הזיכרון של המחשב ב – bytes

`freemem` – כדי לקבל את הזיכרון הפנוי

נבצע חישוב כדי להגיע מ – bytes ל – GB ומכניס את התוצאות לתוך משתנים כמחרוזת תווים

ניצור אובייקט בשם `memoryInfo` נציב אותו כטבלה בטרמינל.

בהפעלת הקובץ אקבל טבלה עם נתוני הזיכרון שהוצאתי בעזרת הספרייה `OS`

```
JS os.js X
CORE_MODULES > JS os.js > ...
1  const OperatingSystem = require("os");
2
3  const totalMemoryInGB = `${OperatingSystem.totalmem() / 1073741824} GB`;
4  const freeMemory = `${OperatingSystem.freemem() / 1073741824} GB`;
5  const memoryInfo = { total: totalMemoryInGB, free: freeMemory };
6
7  console.table(memoryInfo);

PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

C:\Users\DELL\Desktop\HackerU\lecturer-work\Lessons\NODE.JS\NODE.JS-SANDBOX\CORE_MODULES>node os.js

(index)  Values
total    '15.752616882324219 GB'
free     '5.775936126708984 GB'

C:\Users\DELL\Desktop\HackerU\lecturer-work\Lessons\NODE.JS\NODE.JS-SANDBOX\CORE_MODULES>
```

לינק לדוקומנטציה של ספריית OS: <https://nodejs.org/dist/latest-v16.16.0/docs/en/os>

משימת OS



► **הצג בקונסול טבלה עם המפתחות והערכים הבאים:**

► **name** – שם מערכת ההפעלה

► **version** – הגרסה

► **type** – סוג

► **host** – שם המחשב שעליו מותקנת מערכת ההפעלה

► **architecture** – מערכת הסיבים עליה מתבססת מערכת ההפעלה

! לינק לדוקומנטציה של ספריית OS: <https://nodejs.org/dist/latest-x.16/ipa/scod/lmth.so>



ספרייה שמאפשרת לנו גישה
ונייהול ספריות וקבצים במחשב



mkdir

מטודה להוספת ספריות

נשים בתוך קבוע בשם `FileSystem` את האובייקט שיחזור אלינו מהפעלת מטודת `require` של `fs` כאשר נעביר לה בפרמטר את השם

`:mkdir`

המטודה מקבלת בפרמטר בין פרמטר אחד לשלוש כאשר:

בפרמטר הראשון היא מקבלת את הכתובת בה נרצה ליצור את התיקיה

בפרמטר השני - אובייקט קונפיגורציות

בפרמטר השלישי - `call back`

פונקציית ה- `call back` יכולה לקבל בין אפס לשני פרמטרים כאשר:

הראשון הוא אובייקט שגיאה

השני הוא הנתיב שנוצר (כדי להשתמש בפרמטר השני חובה להעביר בקובץ הקונפיגורציות `recursive: true`)

נבצע התניה שאם מתקבלת שגיאה אני עוצר את הפונקציה ומחזיר את הודעת השגיאה

אחרת אני מדפיס את הכיתוב הנ"ל ביחד עם הנתיב שנוצר

```
JS fs.js  X
CORE_MODULES > JS fs.js > ...
1  const FileSystem = require("fs");
2
3  FileSystem.mkdir("./test", { recursive: true }, (error, path) => {
4    if (error) return console.log(error.message);
5    console.log("fs made a file in: \n" + path);
6  });
```

```
PROBLEMS 73 OUTPUT DEBUG CONSOLE TERMINAL JUPYTER
C:\Users\DELL\Desktop\HackerU\lecturer-work\Lessons\NODE.JS\NODE.JS-SANDBOX\CORE_MODULES>node fs.js
fs made a file in:
C:\Users\DELL\Desktop\HackerU\lecturer-work\Lessons\NODE.JS\NODE.JS-SANDBOX\CORE_MODULES\test

C:\Users\DELL\Desktop\HackerU\lecturer-work\Lessons\NODE.JS\NODE.JS-SANDBOX\CORE_MODULES>node fs.js
fs made a file in:
undefined
```

- בפעם הראשונה שאני מבצע את הקוד מכיוון שעוד אין לי תיקייה בנתיב שאני מציין מודפס לי בטרמינל נתיב התיקיה שנוצרה
- בפעם השנייה, מכיוון שיש לי כבר נתיב כזה הוא לא יוצר אותו מחדש ולכן הפרמטר השני ב- `call back` שווה ל- `undefined`

! לינק לדוקומנטציה של מטודת `:mkdir` <https://nodejs.org/docs/latest-v16.x/api/fs.html#fsmkdirpath-options-callback>

rmdir

מטודה להסרת ספריות

:rmdir

המטודה מקבלת בפרמטר בין פרמטר אחד לשלוש
כאשר:

בפרמטר הראשון היא מקבלת את הכתובת בה
נרצה ליצור את התיקיה

בפרמטר השני - אובייקט קונפיגורציות

בפרמטר השלישי - call back

פונקציית ה - call back יכולה לקבל בין אפס
פרמטר אחד של אובייקט שגיאה

נבצע התניה שאם מתקבלת שגיאה אני עוצר את
הפונקציה ומחזיר את הודעת השגיאה

אחרת אני מדפיס את הכיתוב הנ"ל

כאשר אני מריץ את המטודה בפעם הראשונה
היא מזהה שיש תיקייה בנתיב שציינתי, מוחקת
אותה ומציגה לי את הודעת השגיאה

בפעם השנייה שאני מריץ את הקוד המטודה
מזהה שאין תיקייה כזאת בנתיב שציינתי ולכן
מזריקה לפרמטר הראשון בפונקציית ה - call
back את אובייקט השגיאה אותו אני מציג
לקונסול

הודעת שגיאה נוספת יכולה להיות אם אנסה
למחוק תיקייה שיש בתוכה קבצים. במצב זה
הודעת השגיאה תהיה כדלקמן

```
8  FileSystem.rmdir("./test", error => {  
9    if (error) return console.log(error.message);  
10   console.log("directory removed!");  
11  });
```

```
PROBLEMS 73 OUTPUT DEBUG CONSOLE TERMINAL JUPYTER  
  
C:\Users\DELL\Desktop\HackerU\lecturer-work\Lessons\NODE.JS\NODE.JS-SANDBOX\CORE_MODULES>  
node fs.js  
directory removed!  
  
C:\Users\DELL\Desktop\HackerU\lecturer-work\Lessons\NODE.JS\NODE.JS-SANDBOX\CORE_MODULES>  
node fs.js  
ENOENT: no such file or directory, rmdir 'C:\Users\DELL\Desktop\HackerU\lecturer-work\Les  
sons\NODE.JS\NODE.JS-SANDBOX\CORE_MODULES\test'
```

```
PROBLEMS 74 OUTPUT DEBUG CONSOLE TERMINAL JUPYTER  
  
C:\Users\DELL\Desktop\HackerU\lecturer-work\Lessons\NODE.JS\NODE.JS-SANDBOX\CORE_MODULES  
node fs.js  
ENOTEMPTY: directory not empty, rmdir 'C:\Users\DELL\Desktop\HackerU\lecturer-work\Lesso  
ns\NODE.JS\NODE.JS-SANDBOX\CORE_MODULES\test'
```

! לינק לדוקומנטציה של מטודת rmdir: <https://nodejs.org/docs/latest-v16.x/api/fs.html#fsrmdirpath-options-callback>


```

JS fs.js
CORE_MODULES > JS fs.js >  FileSystem.writeFile("writing some text in the file") callback

13  /***** יצירת קובץ *****/
14  FileSystem.mkdir("./test", { recursive: true }, (error, path) => {
15      if (error) return console.log(error.message);
16      console.log("fs made a file in: \n" + path);
17  });
18
19  FileSystem.writeFile(
20      __dirname + "/test/testing.txt", // file
21      "writing some text in the file", //data
22      error => {
23          if (error) return console.log(error.message);
24          console.log("created the testing file in the test folder!");
25      }
26  );

```

```

PROBLEMS 73 OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

C:\Users\DELL\Desktop\HackerU\lecturer-work\Lessons\NODE.JS\NODE.JS-SANDBOX\CORE_MODULES>node fs.js
fs made a file in:
C:\Users\DELL\Desktop\HackerU\lecturer-work\Lessons\NODE.JS\NODE.JS-SANDBOX\CORE_MODULES\test
created a new file in the test folder!

```

```

C:\Users\DELL\Desktop\HackerU\lecturer-work\Lessons\NODE.JS\NODE.JS-SANDBOX\CORE_MODULES>
node fs.js
ENOENT: no such file or directory, open 'C:\Users\DELL\Desktop\HackerU\lecturer-work\Less
ons\NODE.JS\NODE.JS-SANDBOX\CORE_MODULES\test\testing.txt'

```

<https://nodejs.org/docs/latest-v16.x/api/fs.html#fs.writeFile-file-data-options-callback> :writefile של מטודת
 ! לינק לדוקומנטציה של מטודת

writeFile

מטודה להסרת ספריות

שלב ראשון ניצור את תיקיית test

:writeFile

המטודה מקבלת בין שלושה לארבעה פרמטרים:

File - שם הקובץ

Data - מה ייכתב בתוך הקובץ

Option object - אובייקט קונפיגורציות

Call back

:Call back

הפונקציה יכולה לקבל בפרמטר אחד אליו היא תשפוך את אובייקט השגיאה במידה ותהיה אחת כזאת

אעשה התניה ושאם אקבל את אובייקט השגיאה הפונקציה תדפיס אותה בקונסול

אחרת אדפיס שהקובץ נוצר בתיקייה היעד

כאשר אני מריץ את המטודה היא תנסה למצוא את הנתבי שציינתי ואם היא תמצא היא תיצור בתוכו קובץ עם הסיומת שציינתי ובתוכו הכיתוב שהעברתי לה.

במידה והיא לא תמצא את הנתבי שציינתי היא תדפיס לי בקונסול את הודעת השגיאה

unlink

מטודה להסרת קבצים

:unlink

המטודה מקבלת בין שני פרמטרים:

path - נתיב (במקרה הזה השתמשתי באובייקט שמתפקד כגלובלי בסקופ של המודול __dirname כדי לקבל את המיקום האבסולוטי שהקובץ נמצא, אליו אני מחבר את הכתובת הרלטיבית לקובץ שברצוני למחוק

Call back

:Call back

הפונקציה יכולה לקבל בפרמטר אחד אליו היא תשפוך את אובייקט השגיאה במידה ותהיה אחת כזאת

אעשה התניה ושאם אקבל את אובייקט השגיאה הפונקציה תדפיס אותה בקונסול

אחרת אדפיס שהקובץ נוצר בתיקייה היעד

כאשר אני מריץ את הקוד היא תנסה למצוא את הנתיב שציינתי ואם היא תצליח היא תמחק את הקובץ.

במידה והיא לא תמצא את הנתיב שציינתי היא תדפיס לי בקונסול את הודעת השגיאה

```
38  FileSystem.unlink(__dirname + "/test/testing.txt", error => {  
39    if (error) return console.log(error.message);  
40    console.log("file deleted successfully!");  
41  });
```

PROBLEMS 76 OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

```
C:\Users\DELL\Desktop\HackerU\lecturer-work\Lessons\NODE.JS\NODE.JS-SANDBOX\CORE_MODULES>  
node fs.js  
file deleted successfully!
```

```
C:\Users\DELL\Desktop\HackerU\lecturer-work\Lessons\NODE.JS\NODE.JS-SANDBOX\CORE_MODULES>  
node fs.js  
ENOENT: no such file or directory, unlink 'C:\Users\DELL\Desktop\HackerU\lecturer-work\Lessons\NODE.JS\NODE.JS-SANDBOX\CORE_MODULES\test\testing.txt'
```

! לינק לדוקומנטציה של מטודת unlink: <https://nodejs.org/docs/latest-v16.x/api/fs.html#fsunlinkpath-callback>

readdir

מטודה שיוצרת מערך של שמות הקבצים
בנתיב שנועביר לה

:readdir

המטודה מקבלת בין שני פרמטרים:

path - נתיב (במקרה הזה השתמשתי באובייקט שמתפקד כגלובלי בסקופ של המודול __dirname כדי לקבל את המיקום האבסולוטי שהקובץ נמצא, אליו אני מחבר את הכתובת הרלטיבית לקובץ שברצוני למחוק

Call back

:Call back

הפונקציה יכולה לקבל בפרמטר אחד אליו היא תשפוך את אובייקט השגיאה במידה ותהיה אחת כזאת

אעשה התניה ושאם אקבל את אובייקט השגיאה הפונקציה תדפיס אותה בקונסול

אחרת אדפיס שהקובץ נוצר בתיקייה היעד

כאשר אני מריץ את הקוד היא תנסה למצוא את הנתיב שציינתי ואם היא תצליח היא תיצור מערך של שמות הקבצים בנתיב

אחרת היא תדפיס לי בקונסול את הודעת השגיאה

```
68 FileSystem.readdir(__dirname + "/test", (error, files) => {  
69   if (error) return console.log(`Opss... an Error accrued: ${error.message}`);  
70   console.log(files);  
71 });
```

```
C:\Users\DELL\Desktop\HackerU\lecturer-work\Lessons\NODE.JS\NODE.JS-SANDBOX\CORE_MODULES  
>node fs.js  
[ 'testing-0.txt', 'testing-1.txt', 'testing-2.txt' ]
```

PROBLEMS 62 OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

```
C:\Users\DELL\Desktop\HackerU\lecturer-work\Lessons\NODE.JS\NODE.JS-SANDBOX\CORE_MODULES>node fs.js  
Opss... an Error accrued: ENOENT: no such file or directory, scandir 'C:\Users\DELL\Desktop\HackerU\lecturer-work\Lessons\NODE.JS\NODE.JS-SANDBOX\CORE_MODULES\test'
```

! לינק לדוקומנטציה של מטודת unlink: <https://nodejs.org/docs/latest-v16.x/api/fs.html#fsreaddirpath-options-callback>

Asynchronous functions in FS

לרוב המכריע של פונקציות ה - callback
בתיקית FS ישנן הגרסאות אסינכרוניות



async FS

יצירת תיקייה עם קבצים ומערך עם שמות הקבצים בצורה אסינכרונית

נייבא את הפונקציות שמחזירות promise מתוך התיקייה fs/promises

נכניס את הקוד לתוך פונקציה אסינכרונית

ניצור את מנגנון try & catch על מנת שיתפוס שגיאות במידה והיו

נפעיל את מטודת mkdir שתיצור לנו תיקייה בנתיב שהעברנו לה. נשתמש במילה השמורה await כדי ליצור מעין סינכרוניות בקוד כך שרק לאחר ביצוע המטודה הקוד ימשיך, אלא אם תיזרק שגיאה מהפונקציה

ניצור לולאה שתיצור שלוש קבצים בעזרת המטודה האסינכרונית writeFile אליה נעביר את הנתיב (שכולל גם את שם הקובץ) ומה ייכתב בתוך הקובץ שהיא תיצור, ולבסוף נדפיס את מחרוזת התווים

ניצור קבוע בשם filesArray שערכו יהיה המערך שיחזור אלינו מהפעלת הפונקציה readdir האסינכרונית

נדפיס את המערך שנוצר בקונסול.

התוצאה בטרמינל

```
71 const { mkdir, readdir, writeFile } = require("fs/promises");
72
73 const fn = async () => {
74   try {
75     await mkdir(`${__dirname}/test`);
76
77     for (i = 0; i < 3; i++) {
78       await writeFile(__dirname + `/test/testing-${i}.txt`, `file no. ${i}`);
79       console.log(`testing-${i}.txt has been created!`);
80     }
81
82     const filesArray = await readdir(__dirname + "/test");
83     console.log(filesArray);
84   } catch (error) {
85     console.log(error.message);
86   }
87 };
88
89 fn();
```

PROBLEMS 63 OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

```
>node node fs.js
testing-0.txt has been created!
testing-1.txt has been created!
testing-2.txt has been created!
[ 'testing-0.txt', 'testing-1.txt', 'testing-2.txt' ]
```

.then() FS

מחיקת התיקייה והקבצים שבתוכה בצורה
אסינכרונית

▶ נייבא את הפונקציות שמחזירות promise
מתוך התיקייה fs/promises

▶ נפעיל את המטודה האסינכרונית readdir
ונעביר לה בפרמטר את הנתיב של התיקייה

▶ נפעיל את מטודת .then

▶ תשפוך את המידע לפונקציה האנונימית
שבתוכה לתוך המשתנה files

▶ תפעיל את מטודת forEach על מערך הקבצים
כך שעל כך איבר במערך היא תפעיל את
המטודה האסינכרונית unlink שתסיר את
הקובץ.

▶ נפעיל שוב את מטודת .then

▶ נפעיל בתוכה את המטודה האסינכרונית
rmdir להסרת התיקייה

▶ אם תהיה שגיאה באיזה שהוא שלב נתפוס
אותה עם המטודה .catch

```
93 const { readdir, rmdir, unlink } = require("fs/promises");
94
95 readdir(__dirname + "/test")
96   .then(files => files.forEach(file => unlink(`${__dirname}/test/${file}`)))
97   .then(() => {
98     rmdir(`${__dirname}/test`).catch(console.log);
99   })
100   .catch(console.log);
```

משימת FS



- ▶ ייבא את האובייקט מתיקיית fs לתוך קבוע
- ▶ ייבא את הפונקציות הבאות מתוך תיקיית fs/promises:
 - ▶ mkdir, readdir, writeFile, rmdir, unlink
- ▶ צור קבוע בשם users שהערך שלו יהיה מערך שיכיל שלושה אובייקטים שייצגו משתמשים ובכל אובייקט שיהיה את המפתח name ואת המפתח last
- ▶ צור פונקציה אסינכרונית בשם removeFilesAndFolder:
 - ▶ שתיצור קבוע בשם users שיכיל את מערך שמות הקבצים שבתוך התיקייה users
 - ▶ שתמחק את כל הקבצים הללו
 - ▶ ולבסוף תמחק את התיקייה users
- ▶ צור פונקציה אסינכרונית בשם makeAndRemoveFilesAndFolder שבתוכה
 - ▶ בדוק בעזרת המטודה המתאימה אם הנתיב __dirname + '/users' קיים ואם כן שתפעיל את הפונקציה removeFilesAndFolder
 - ▶ ואם הנתיב לא קיים:
 - ▶ צור תיקייה בשם users
 - ▶ צור ארבעה קבצי txt בתוכה כאשר כל קובץ יהיה מורכב מהשם הפרטי של המשתמש, מקף ושם המשפחה ובתוך כל קובץ יהיה כתוב שמו של המשתמש
- ▶ הפעל את מטודת setTimeout שכעבור חמש שניות תפעיל הפונקציה removeFilesAndFolder



ספרייה המאפשרת לנו גישה
לשורת הכתובת של קבצים
ותיקיות או לחלקים ממנה.



basename

מטודה לקבלת שם הקובץ

נשמור את האובייקט שיחזור מהפעלת
מטודת require עם הערך path בתוך
קבוע בשם path

נדפיס בטרמינל את הפעלת מטודת
path.basename ונעביר לה בפרמטר
כתובת.

התוצאה: הדפסת שם הקובץ בטרמינל

JS path.js X

CORE_MODULES > JS path.js > ...

```
1  const path = require("path");  
2  
3  console.log(path.basename(`${__dirname}/path.js`));
```

```
C:\Users\DELL\Desktop\HackerU\lecturer-work\Lessons\  
NODE.JS\NODE.JS-SANDBOX\CORE_MODULES>node path.js  
path.js
```


JS path.js X

CORE_MODULES > JS path.js > ...

```
1  const path = require("path");  
2  
3  console.log(path.join(__dirname, "/path.js"));
```

```
C:\Users\DELL\Desktop\HackerU\lecturer-work\Lessons\  
NODE.JS\NODE.JS-SANDBOX\CORE_MODULES>node path.js  
C:\Users\DELL\Desktop\HackerU\lecturer-work\Lessons\  
NODE.JS\NODE.JS-SANDBOX\CORE_MODULES\path.js
```

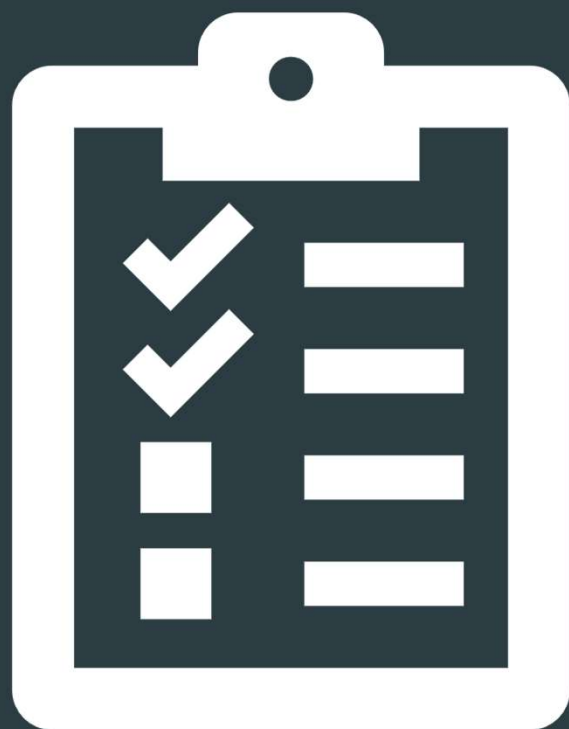
join

מטודה שמאחדת חלקי כתובת

נדפיס בטרמינל את הפעלת מטודת path.join ונעביר לה בפרמטר את כתובת התיקייה ובפרמטר השני את שם הקובץ.

התוצאה: הדפסת שם הכתובת המלאה בטרמינל

משימת PATH



- ▶ בהמשך לתרגיל הקודם של משימת FS
- ▶ התנה שרק אם כל הקבצים בתיקייה למחיקה הם עם סיומת .txt. התיקייה תימחק
- ▶ אחרת רק הקבצים עם סיומת .txt. ימחקו אבל התיקייה תישאר עם הקבצים עם הסיומות האחרות
- ▶ בדוק את המטודה המחודשת כאשר תכניס לתיקייה של users קובץ עם סיומת של .pdf.