**NodeJS:**

**התקנה**:

1. נוריד את קובץ ה-tar שלנו מהאתר <https://nodejs.org/en/download/>. נביא אותו למכונת הלינוקס שלנו למיקום שיש לנו הרשאות עליו.
2. נבצע tar –xvf ונחלץ את התיקייה שנמצאת בקובץ הזה.
3. זוהי התיקייה של ה-NodeJS, ניתן להשתמש גם בה.

דרך שנייה:

1. נריץ את הפקודה:

curl -sL https://deb.nodesource.com/setup\_5.x | sudo -E bash –

1. לאחר מכן נריץ את הפקודה:

sudo apt-get install -y nodejs

**ויש לנו NodeJS!**

|  |  |
| --- | --- |
| **פירוש** | **פקודה** |
| בדיקת גרסה | node --version |
| להריץ את פקודת ה-node | node file\_name |
| בדיקת האתר | curl 127.0.0.1:8080 |
| Server לדוגמא  שורה ראשונה- הכלל המודול של http, ושמירתו כמשתנה בשם http.  כל השאר די ברור... | var http = require("http");  http.createServer(function(request, response){  response.writeHead(200, {"Content-Type": "text/html"});  response.end("Hello World!");  }).listen(8080); |
| Import the MySQL module.  Create Connection to DB.  Execute query.  Execute SQL Transaction.   1. Insert 2. Update 3. Delete   Terminate connection gracefully. | var mysql = require("mysql");  var con = mysql. createConnection({  host : "localhost",  user : "root",  password : "Password1",  database : 'general'  });  con.connect(function(err){  if(err){  console.log('Error connecting to DB');  return;  }  console.log('Connection established');  });  con.query('SELECT \* FROM enteries' , function(err , rows){  if(err) throw err;  console.log(rows)  });  var entery = { ip : '127.0.0.1' , browser : 'Mozilla' , time : 'CURRENT\_TIMESTAMP'};  con.query("sql\_query\_string", function(err , res){  if(err) throw err;  console.log('Last insert ID:' , res.insertID);  });  con.end(function(err){}); |
| Get User IP Address | npm install request-ip --save  var requestIp = require('request-ip');  In the connection function:  var clientIp = requestIp.getClientIp(req);  //IPV4  clientIp = clientIp.substring(7); |
|  |  |

לראות את ההכנסות, נרצה שהטבלה שלנו תבצע log החוצה של השאילתות. 2 דרכים:

1. לתוך טבלה של MySQL:

SET GLOBAL log\_output = 'TABLE';

SET GLOBAL general\_log = 'ON';

וצריך להסתכל בטבלה mysql.general\_log.

1. לתוך קובץ:

SET GLOBAL log\_output = 'FILE';

SET GLOBAL general\_log\_file = '/path';

SET GLOBAL general\_log = 'ON';

וצריך להתסכל בקובץ שיצרנו.

**npm**- A node.js package manager.

**NodeJS- From Zero to Hero:**

**What is NodeJS?**

NodeJS is a runtime environment and library for running JavaScript application outside the browser.

NodeJS is mostly used to run real-time server application and shines through its performance (מעולה בביצועים) using non-blocking I/O and asynchronous events.

**When to use node:**

1. Chat applications
2. Real time applications and collaborative (שיתופי) environment
3. Game servers
4. Ad servers
5. Streaming (נהירה, הזרמה) servers

Good with a high level of concurrency (מקביליות) and a little CPU time.

**Not good for:**

Long time operations.

**1) Hello World:**

console.log(“Hello World!”);

**2) HTTP Hello World:**

var http = require('http');

var server = http.createServer(function(request, response){

response.writeHead(200, {'Content-type': 'text/plain'});

response.end('Hello World!');

});

server.listen(7000);

**Event Driven Programming**- A programming paradigm in which the flow of the program is determined by events such as user actions (mouse clicks, key presses) or messages from other other programs.

**3) Read file- Blocking IO:**

var fs = require('fs');

**//Read file- this block all the other operations!! Stoping the web server, if exist!!**

var contents = fs.readFileSync('package.json').toString();

console.log(contents);

**//Better way- Syncronized!**

fs.readFile('package.json', function(err, buf){

console.log(buf.toString());

});

**Callback Style Programming**-

* Event loops result in callback-style programming where you break apart a program into its underlying (בסיסי) data flow.
* In other words, you end up splitting your program into smaller and smaller chunks until each chunk is mapped to operation with data.
* Why? So that you don't freeze the event loop on long-running operations (such as disk or network IO).

**4) Read multiple files- Callback insanity!**

var fs = require('fs');

fs.readdir('.', function(err, files){

if(err){

console.log('Error finding files:' + err);

}

else

{

files.forEach(function(fileName, fileIndex){

fs.readFile(fileName, function(err, buf){

if(err)

{

console.log('Error reading file:' + err);

}

else

{

console.log(buf.toString());

}

});

});

}

});

**Promises:**

* A function will return a promise for an object in the future.
* Promises can be chained together
* Simplify programming of async (לא סינכרוני) systems.

<https://www.youtube.com/watch?v=czmulJ9NBP0>

<https://github.com/sayar/NodeMVA/tree/master/04_CallbackInsanity>