

Node.js

Date / / No



cmd → node -v → ✓  
→ cls → clear cmd

node → console.log (process.exit())  
Process → process.exit()

Blocking

vs

Non Blocking

const user = getUser(1)  
console.log(user);

getUser(1, (user) => {  
 console.log(user)  
})

the Fast

Node.js uses event driven, non blocking I/O model that makes it lightweight and efficient

Section 3

\* You should import fs → const fs = require('fs')

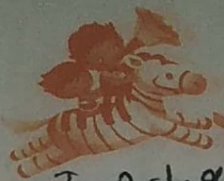
fs.writeFileSync('notes.txt', 'any name you want')

↓  
or node app.js → run the code

\* افری انت عالم کذا ملک وناوز تنوی output یاغی  
require('fs')  
node app.js  
output → app output

make.export = {}





Date / / No

## Importing npm Modules

npm init → package.json → Package JSON  
↓  
Data Script object notation

npm → Validator Package → npm i validator  
↳ chalk → npm i chalk → ES6 type: module

## Getting input from user

node app.js add

add → array

→ console.log(process.argv)

↓  
[ --title, add ]

node app.js --title = "عنوان" ←

→ npm i yargs → const yargs = require("yargs");

yargs: Command({  
 command: "add",  
 describe: —

builder: {

title: {

}



Date / / No

## Storing data with JSON

JSON.stringify (object, array)

to convert object or array to JSON

JSON.parse ( )

to convert from JSON to object

## debugging Node.js

(1) console.log → Parameters في الموضع في الذاكرة

(2) debugger → in terminal → node inspect app.js  
↓  
inspect - brk

edge : // inspect /x devices  
Chrom :

Folder → Add Folder to work space

debugger → run → [F5] (تشغيل)

restart → console VSC → أكتب في

Ctrl + C → توقف → توقف عن العمل



Asynchronous Basics

Ex

```

console.log('Start 1')
setTimeout(() => {
  console.log('Start 2')
}, 2000)

setTimeout(() => {
  console.log('Start 3')
}, 0)

console.log('Start 4')

```

out Put

Start 1

Start 4

Start 3

Start 2

Call Stack

↳ JS

Node API

↓

Call Queue

↓

applying Call Stack

Call back function → is a function we provide it as an argument to another function

```

setTimeout(() => {
  console.log('')
}, 3000)

```

Object destructuring

```

const Product = {
  label: 'Red notebook',
  Price: 3,
  Stock: 201,
  SalePrice: undefined
}

```

const label = Product.label  
 ~ Stock = ~. Stock

const {label, Stock} = Product



url / Products ? Search = James

Date / / No

## Query String:

The client, when he write the url should put query string then the server will use this data and send response back

## heroku

heroku create  
git remote →  
git push heroku main

heroku  
origin →

وکت کام لایو  
نویسای تغییر

git status  
git add  
git push

git commit -m "

git push heroku main

## MongoDB

/Users/softzone/mongodb/bin/mongod.exe  
--dbpath=/Users/softzone/mongodb/data

نویسای تغییر

## REST API

Representational State Transfer - Application Programming Interface



## Promise chaining

### Promise

```
const add = (a, b) => {
  return new Promise((resolve, reject) => {
    set timeout (1) => {
      resolve (a + b)
    }, 2000)
  })
}
```

### Promise chaining

```
add (1, 2).then (Sum) => {
  console.log (Sum)
  return add (Sum, 4)
}).then (Sum2) => {
  console.log (Sum2)
}.catch (e) => {
  console.log (e)
}
```

### Async/Await

Async function return Promise

It is readable with  
 async and await  
 and you can use all sums  
 but Promise chaining you can't  
 use a lot of one sum in  
 then()

```
const doWork = async () => {
  const sum = await add (1, 99)
  ~ ~ 2 = ~ ~ (Sum, 50)
  ~ ~ 3 = ~ ~ (Sum2, 3)
  return sum3
}
```



Date / / No

npm bcryptjs → bcrypt, hash (Password, 8)

↓

npm jsonwebtoken

npm Pagination

npm multer → for uploading files

↳ npm sharp → for images → crop and others

\* [Sandgrid.com](https://sandgrid.com)

Testing

, Super test

, Mocha → JS test framework running on Node.js

, Jest → ✓

↳ npm i jest

① "test" : "jest" → Package configuration

② make new file with → name.test.js

Why test → Save time

↳ create reliable software

↳ gives flexibility to developers, Refactoring...