Node JS

في هذا الكورس سوف اشرح
مفاهيم اكثر من شرح مفاهيم بدون تطبيق
عملي.

مفاهيم => 50% استيعاب.
مفاهيم + تطبيق => 100% استيعاب.

محمد أمير

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

Wednesday

5

May

٥

مـــايـــو

٢٣ رمضان ١٤٤٢ هـ

الأربعاء

٢٧ برموده ١٧٣٧ ق

MongoDB

٧

٨

٩

١٠

١١

١٢

١

٢

٣

٤

٥

٦

٧

https://www.linkedin.com/in/mohamadamir/

الممسوحة ضوئيا بـ CamScanner

Tuesday

٤

4

مـــايـــو

May

الثلاثاء

٢٢ رمضان ١٤٤٢ هـ

٢٦ برمـوده ١٧٣٧ ق

1- intro to NoSQL database.
2- intro to MongoDB.
3- get started.
4- crud
5- Data Models
6- Aggregation
7- ملاحظات مهمة

11

12

1

2

3

4

5

6

7

Sat. 1 Sun. 2 Mon. 3 Tue. 4 Wed. 5 Thu. 6 Fri. 7 Sat. 8 Sun. 9 Mon. 10 Tue. 11 Wed. 12 Thu. 13 Fri. 14 Sat. 15 Sun. 16 Mon. 17 Tue. 18 Wed. 19 Thu. 20 Fri. Sat. Sun. Mon. Tue. Wed. Thu. Fri. Sat. Sun. Mon.

https://www.linkedin.com/in/mohamadamir/

الممسوحة ضوئيا بـ CamScanner

① intro to NoSQL database :-

Q) ايه هو الـ database ؟

هي مجموعة ملفات بتخزن فيها الـ data بشكل منظم بحيث نقدر نـ manage الـ data دي بسهولة.

Q) ايه انواع الـ database ؟

١- SQL database / Relational database ، وهي الـ db اللي بتخزن الـ data على شكل جداول فيها علاقات ما بينهم.

٢- NoSQL db / object-oriented db ، وهي الـ db اللي بتخزن الـ data على شكل objects ومبتحترمش مبدأ العلاقات.

Q) ايه انواع الـ data اللي بتتعامل معاها ؟

١- structured data :- دي الـ data اللي ليها structure واضح نقدر نمثلها في tables ونقدر نعمل analysis بسهولة.
زي مثلاً) بيانات الـ clients / بيانات منتج / ....

٢- unstructured data :- دي الـ data اللي ملهاش format واضح يعني مقدرش نمثلهم في جداول وصعب جداً نحللهم ، يعني الـ data اللي ملهاش schema ثابتة.
زي مثلاً) الصور ، الفيديوهات ، الصوت ، بوستات الـ facebook ، الـ emails ، ....

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

Q) ايه هو الـ SQL؟

هي الـ query language المسئولة عن التعامل مع الـ RDBMS
هي مجموعة أوامر/ queries بنتكلم بيها الـ relational databases.

Q) ايه هو الـ NoSQL؟

دة نوع من انواع الـ DBMS اللي بيخزن البيانات على شكل objects وليس جداول بالتالي مبيحتاج الـ relationship
ومناسب جدا لتخزين الـ unstructured data.

كلمة noSQL المقصود بيها " not only SQL " يعني مش هيستبدل الـ SQL database ولكن هيوفر options جديدة لإدارة الـ data.

Q) ايه اهم مميزات الـ NoSQL database؟

١- البنية بتاعته هي schemaless يعني ملهوش schema ثابتة بالتالي نقدر نغير من عدد الـ fields بتاعته بسهولة اثناء الـ runtime.
٢- مناسب لتخزين الـ unstructured data.
٣- مناسب لإدارة كم مهول جدا من الـ data.
٤- نقدر نقسم/ نوزع الـ noSQL database على أكثر من server.
٥- مناسب مع الـ system اللي ملهوش schema ثابتة.

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

٧) ايه الفرق بين الـ NoSQL db والـ SQL db؟
٢) امتى نستخدم الـ NoSQL والـ SQL؟

١- الـ SQL db بيخزن الـ data على شكل جداول.
الـ NoSQL ~ الـ ~ ~ ~ JSON / objects.
على حسب نوع الـ NoSQL db اللى هنستخدمه.

٢- الـ SQL db هو vertical scalable يعني يقدر يزود أو يقلل
عدد الـ rows لكن صعب جداً يعدل في عدد الـ columns بالتالي
هو schemafull يعني ليه schema ثابتة وواضحة.

٣- الـ NoSQL هو horizontally scalable يعني يقدر يعدل في
عدد الـ columns / الـ fields اللى فيه بسهولة جداً بالتالي
هو schemaless.

٣- الـ SQL db سهل جداً نبعت ليه queries معقدة.
الـ NoSQL صعب جداً ~ ~ ~ ~.

٤- لو عندنا حجم مهول جداً من الـ data مع إستخدام الـ relations
ده هيأثر بشكل واضح على الـ performance.

٥- الـ SQL db صعب توزعه على أكثر من server.
الـ NoSQL سهل ~ ~ ~ ~ ~.

٦- الـ SQL db مناسب جداً لتخزين الـ structured data
الـ NoSQL ~ ~ ~ ~ الـ ~ unstructured

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

٧- الـ SQL db مناسب للمشاريع اللي بتحتاج schema ثابتة.
الـ noSQL ~ ~ ~ ~ dynamic schema
يعني المشاريع اللي مالهاش schema واضحة.

٨- الـ noSQL مناسب أكثر مع المشاريع اللي بتحتاج تـ manage كم مهول جداً من الـ data

٩- الـ SQL مناسب أكثر مع المشاريع اللي بتحتاج تشكل relations بين الـ data.

١٠- الـ SQL db بتستخدم نوع واحد من الـ query language لأكثر من DBMS.

١١- الـ noSQL بتنقسم لأكثر من نوع وكل نوع ليه الـ DBMS الخاص بيه والـ query language الخاص بيه، لكن الميزة إن أغلبهم free و open source.

١٢- الـ noSQL مبيستخدمش الـ structural query language إنما بيستخدم query language خاص بيه. وده بيعتمد على نوع الـ noSQL db اللي هنستخدمه.

١٣- أنواع الـ noSQL db هي:
1- Key-value db
2- column-based db
3- graph db
4- object-oriented db
5- document db ⟵ أشهر أنواع الـ noSQL

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

② Intro to MongoDB :-

@) ايه هو الـ MongoDB ؟

هو عبارة عن db no SQL document بيخزن الـ data على شكل object / document / json .

البنية بتاعة الـ MongoDB عبارة عن مجموعة collections وكل collection بيتكون من مجموعة documents وكل document بيتكون من مجموعة fields .

الـ collection شبه الـ table في الـ RDBMS
الـ document ~ الـ row / record ~ ~
الـ field ~ الـ column في الـ ~ ~

١- الـ db SQL بتستخدم fixed schema يعني الـ structure بتاع الـ tables لازم يتعمل الأول قبل ما نحط فيه الـ data .

٢- الـ Mongodb بيستخدم dynamic schema يعني كل document ممكن يحتوي على field مختلفة عن التاني خالص .

٣- الـ Mongodb هي document-oriented يعني البنية الاساسية أو الوحدة الاساسية للـ Mongodb هي الـ document .

٤- الـ Mongodb اتعمل عشان يعمل scalable db ، يعني db سهل تتعدل / تتغير .

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

@) امتى من المناسب نستخدم MongoDB ؟

١- مع الـ web application اللى بتحتوى على كم كبير جداً من الـ data وبتحتوى على unstructured data .

٢- مع المشاريع اللى ملهاش schema ثابتة وواضحة .

٣- مع المشاريع اللى بتحتاج real-time يعنى بتحتاج سرعة عالية فى تخزين وارجاع الـ data .

٤- مع المشاريع اللى بتحتاج scalability عالية .

@) امتى الأفضل نستخدم MongoDB ؟

١- مع الـ applications اللى بتحتاج تعمل transaction كتير ، الـ transaction عبارة عن مجموعة من الـ db operations اللى هيتنفذوا مع بعض كـ unit واحدة ، بالتالى لازم ينفذوا كلهم مع بعض أو يتلغوا كلهم مع بعض ، ميتنفذش حاجة وحاجة لا ؛ زى مثلاً) عمليات تحويل الفلوس فى البنك .

٢- لو بتحتاج تعمل complex queries كتير فى الـ application ؛

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

Q) هل نقدر نعمل relations في الـ MongoDB ؟
Q) ~ ~ ~ transaction ~ الـ ~ ؟

* الـ MongoDB هي مش relational database لكن نقدر نعمل relations ولكن الـ process اللي هنعملها هتكون أطول واصعب من إن إحنا نعملها بالـ relational dbms.

* الـ MongoDB بيقدر يعمل transaction لكن الـ process اللي هنعملها هتكون اصعب من إن إحنا نعملها بالـ RDBMS.

Q) ازاي الـ MongoDB بتشتغل ؟

اولاً لازم نعرف المصطلحات المهمة اللي هنقولها كتير.

in MongoDB	in RDBMS
Database	Database
Collection	Table
Document	record / row
field	column
Embedded Documents	Table join
primary key (_id key)	primary key

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

Q) ايه الفرق بين ال JSON و ال BSON ؟

ال BSON هو نوع من انواع ال JSON الفرق إن :
① ال Json بيتخزن على شكل text إنما ال BSON بيتخزن على شكل binary format بالتالي هو اسرع من ال Json.

② ال BSON بيخزن datatype جهاز أكثر من ال json.
ال json بيخزن 6 انواع بس من ال data اللي هما
(null، array، object، boolean، number، string).

الخلاصة إن ال BSON هو ال JSON لكن بيتخزن في ال hard desk على شكل binary code، وبيدعم datatype جهاز. زي ال Date و ال binary array.

Q) ايه هي ال query language الخاصة بال MongoDB ؟

ال MongoDB query language (MQL) ودية اللغة المسئولة عن التعامل مع ال data اللي موجودة في ال MongoDB.

اللغة دية هي مش human friendly / مش declarative زي ال SQL للاسف، لكن ال syntax بتاعها يعتمد على ال JSON عشان تومر queries مناسبة للتعامل مع ال data اللي في ال MongoDB.

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

③ get started :-

@) ازاى تبدا تستخدم الـ MongoDB ؟

1- اولاً لازم تكون مجهز الآتى :

1- لازم تكون منزل الـ MongoDB بالجهاز .
2- لازم ~ ~ الـ MongoDB shell بالجهاز .
2- اعمل folder جوا الـ directory c:// باسم data/db
↳ c:\data\db

دى عبارة عن الـ default directory اللى MongoDB بيخزن فيه الـ data .

2- ثانياً لازم نشغل الـ server بتاع الـ MongoDB .

1- هتروح للمكان اللى الـ MongoDB نزل فيه ، تحديدا المكان اللى فيه الـ (mongod.exe) .
2- من هناك افتح الـ cmd .
2- اكتب امر (mongod) .

```
>> cd c:\Program Files\MongoDB\Server\6.0\bin ↵
>> mongod ↵
```

3- ثالثاً هتروح للمكان اللى فيه الـ mongoDB shell وتشغله run .

```
>> cd c:\mongosh\bin
>> mongosh ↵
```

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

* قبل ما تبدأ أي حاجة لازم تكون مشغل حاجتين :

① run MongoDB daemon (mongod)
② run MongoDB shell (mongosh)

1- To create DB (or) select exist DB:

```
>> use mydb ↵
```

لو mydb موجود، هيعمل select عشان نشتغل عليه.
لو مش موجود، هيعمل create.

2- عشان تشوف ايه الـ db اللي إنت عامل select
يعني تشوف ايه الـ db اللي إنت واقف عليها :

```
>> db ↵
```

3- عشان تشوف ايه كل الـ db اللي عندك :

```
>> show dbs ↵
```

لازم الـ db يكون فيه collection واحد على الاقل
عشان تظهر في الـ list.

4- عشان تعمل drop للـ db اللي إنت عامل select.

```
>> db.dropDataBase() ↵
```

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

④ Basic MongoDB operations :-

①
```
>> db.createCollection("blz") ↵
>> db.blz.drop() ↵
>> show collections ↵
```

② insert

ازاى نعمل insert لـ document فى الـ collection.

```
>> db.blz.insert({name:'teto'}) ↵
```

اسم الـ collection — الـ document

③
فى الـ mongoDB مش لازم نعمل create للـ collection بنفسنا، لأن لما نعمل insert لـ document الـ MongoDB هيعمل create للـ collection بنفسه لو مش موجود.

④ get operation

ازاى نعمل get لكل الـ documents فى الـ collection

```
>> db.blz.find() ↵
>> db.blz.find().pretty() ↵
>> db.blz.find({title:'node'}).pretty() ↵
```

الـ condition

```
>> db.blz.find({title:'KoK'}, {title:1, bt:1}).pretty() ↵
```

الشرط اللى هنعمل عليه

اية الـ fields اللى عايزها تتعرض)
قيمة 1 يعنى true يعنى عايزه يتعرض.

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

find

```
>> db.blz.find( الشرط , ايه اللي بتعرض )
>> db.blz.find({}, {Key1:0, Key2:1})
```

conditions — Logical operators

==	{name: 'Ali'}
<	{age:{$lt:18}} less than
<=	{age:{$lte:18}} less than or equal
>	{age:{$gt:18}}
>=	{age:{$gte:18}}
!=	{age:{$ne:18}}

and	{$and:[{K1:V1},{K2:V2}]}
or	{$or:[{ ↓ },{ },{ }]}

الشرط

find

```
>> db.blz.findOne( , )  رجع أول result فقط
>> db.blz.find().limit(100)  رجع اول 100
>> db.blz.find().limit(100).skip(10)  عدى أول 10 ورجع الـ100 اللي بعدهم
```

sort

```
>> db.blz.find().sort({KEY:1})
```

تصاعدي ← 1
تنازلي ← -1

update

```
>> db.blz.update({ ↓ }, { ↓ }, {multi:true})
```

الشرط / الفلتر — الحاجة اللي هتتعدل

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

delete

>> db.blz.remove({ })

الشرط ، لو الشرط فاضي
كده هيمسح كله .

(ملحوظة)
* عشان تخلي run للـ MongoDB server بشكل تلقائي :
① هتفتح برنامج الـ services بتاع الـ OS .
② هتدور على الـ MongoDB server .
③ اعمله start / run

روح اقرا الـ document عشان تعرف تفاصيل أكثر : mongodb.com/docs/manual/crud/

Q) ايه هو الـ db object ؟

---

* ده عبارة عن global object بيتعمل أول ما الـ client يعمل connect & MongoDB database ،

* الـ db object بيمثل الـ MongoDB database اللي احنا عاملين عليها connect .

* الـ db object بنستخدمه عشان نوصل للـ collections والـ document اللي جوا الـ db ونعمل عليهم queries أو crud operations .

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

```
db.users.find(                    → collection ال
  {age:{$gt:18}},                 → query criteria ال
  {name:1, city:1}                → projection ال
).limit(5)                        → cursor modifier ال
```

- ال query criteria هو الشرط اللى هنفلتر بيه ال result.
- ال projection هى ال fields اللى عايزينها تظهر فقط.
- ال modifier هو التعديل اللى هنعمله على ال result.

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

(5) Data Models (relations, referencing) :-

١- الـ data modeling ممكن نقول عملية الـ relations ما بين الـ documents ، لكن إحنا مش بنستخدم RDBMS بالتالي بنستخدم مصطلح الـ data modeling وهي عملية تشكيل الـ structure بتاع الـ data.

٢- عندنا نوعين من الـ relations / الـ data modeling :

① embedded data :

← دي عبارة عن document واحد بيحتوي على الـ documents اللي ليهم علاقة ببعض .

② referencing :

← دي عبارة عن إن إحنا بنفصل كل document لوحده وبنربط بينهم بالـ reference اللي هو شبه الـ FK .

• عيبها إنه أصعب في التخزين وبنحتاج نعمل query معقدة عليه .

* لو الـ nested document قليلة ومش هتسبب تكرار كثير للـ data من غير داعي ، يبقى يفضل نستخدم الـ embedded .

* لو الـ nested data كثيرة جداً وفيه تكرار للـ data ، إذا يفضل نستخدم الـ reference .

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

لما نعمل reference relation ما بين 2 documents
هنستخدم 3-fields اساسيين :

① $ref ← بيمثل اسم ال collection اللي
هنعمل منه ال document ال reference .

② $id ← هيحتوي على قيمة ال _id بتاع ال document
اللي هنعمل reference عليه .

③ $db ← هيحتوي اسم ال database اللي شايل
ال document اللي هنعمل reference عليه

(ex) Emp

```
{_id: 10,
 name: "ali",
 car: {id: 20,
       color: "red",
       model: "verna"
      }
}
```

ديه طريقة ال embedded

```
car: { $ref: 'Car',
       $id: 20
     }
}
```

ديه طريقة ال reference

Car

```
{_id: 20,
 color: "red",
 model: "verna"
}
```

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

⑥ Aggregation

* الـ aggregate هو إن إحنا نطلع إحصائية من الـ collection زي حساب مجموع field معين لكل الـ documents اللي ليهم نفس الـ field معين.

(ex)

```
>> db.blz.aggregate([
    { $match : {} },
    { $group : { _id : "$name", total : { $sum : "$_____" } } }
])
```

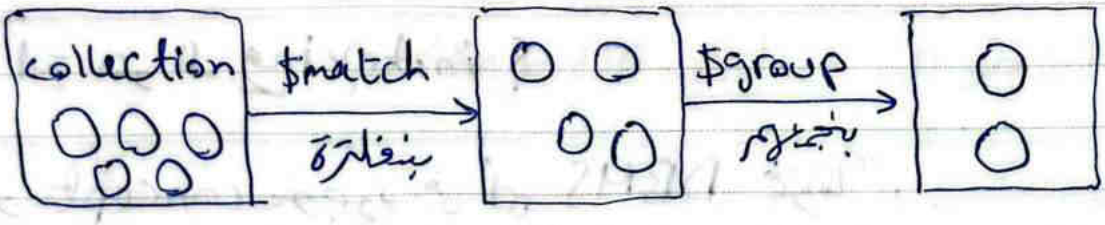
الشرط اللي هنفلتر بيه.

اية الـ field اللي هنعمل بيه grouping.

اية الـ field اللي هنعمل فيه aggregate.

اية الـ aggregate اللي هننفذه

* عشان كده إحنا بنمشي بمرحلتين في الـ grouping :-



* لو هنستخدم aggregate جاهز، يفضل نستخدم الـ grouping.

* لو عايزين نعمل custom aggregate، يفضل نستخدم الـ (map - reduce) وديه طريقة advanced للـ grouping العادي.

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

ملاحظات ⑦

Q) الـ MongoDB هي nosql db يعني schemaless، لكن هل نقدر نعمل static schema لـ collection؟

اه ينفع نعمل static schema باستخدام الـ schema validation
وإحنا بنعمل create لـ collection، الـ collection هو شبه الـ table في الـ RDBMS.

ex) db.createCollection("students", {validator:{ }})

لـ الـ schema بتاعة لـ اسم الـ collection الـ documents، يعني
شكل الـ documents اللي هيتحط في الـ collection ده.

روح لـ MongoDB document واقرا عن الـ schema Validation

Q) ايه هو الـ indexing؟

- هو concept موجود في أي DBMS عموماً.
- الـ index هو حاجة الـ DBMS بتحطه في كل record عشان تسرع عملية الـ search عملية، لكن عيبه بيبطأ عملية الـ insert والـ update والـ delete.
- إحنا بنحدد الـ field اللي بنستخدمه في الـ search وبنحط عليه index.

ex) db.blz.createIndex({name: -1})

الترتيب بتاع الـ indexes (تصاعدي، تنازلي)

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

Q٧) هل نقدر نعمل backup و restore في الـ MongoDB؟

← بنعمل باستخدام tool تبع الـ MongoDB اسمها mongodump.exe و mongorestore.exe.

← ازاي نعملهم install:
① هتروح لمكان الـ download.
② هتروح لـ tools
③ هتروح لـ MongoDB database tools واعمل install.
④ هتفتح الـ folder اللي نزل وتاخد منه الـ mongodump.exe والـ mongorestore.exe وتحطهم في folder الـ bin اللي فيه الـ mongod.

>> mongodump --db=biz ↵
كده عملنا backup لـ biz database.

>> mongorestore + backup path ↵

Q٢) ايه هو الـ MongoDB Atlas؟

هو نفس الـ MongoDB لكن مرفوع على الـ cloud، يعني بدل ما نشتغل على الـ database بتاعتي local، هنشتغل عليها من الـ cloud، الـ cloud server هيدينا الـ url للـ db بتاعتنا عشان نقدر نـ connect عليها مش الـ local.

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

Tuesday

13

April

١٣

ابريل

١ رمضان ١٤٤٢ هـ

الثلاثاء

٥ برموده ١٧٣٧ ق

Node JS

https://www.linkedin.com/in/mohamadamir/

الممسوحة ضوئيا بـ CamScanner

Node.js

Session (1)

1- intro to Node js.
2- Node js architecture.
3- modules, package manager.
4- Node JS webserver.

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

① Intro to Node js

Q) ايه هو ال nodejs؟

ال nodejs عبارة عن runtime environment :js مبنية على ال engine الخاص بـ Chrome (V8) js، وبالاضافة بتحتوي على core Modules بتمكننا نعمل web server بال js، ونقدر نتعامل ال filesystem بال js، ونقدر نتعامل مع ال database بال js.

Q) ايه الغرض من ال node js؟

إن إحنا نعمل application js بيشتغل خارج ال browser، ونقدر نعمل server بال js.

Q) ايه هو ال runtime؟ هي الحاجة اللي بتشغل الكود عالجهاز.

Q) ليه نستخدم node js؟ امتى نستخدمه؟

ال node js بيعمل applications سريعة جداً.
ال " يقدر يبني web server بوقت قليل.
ال " مناسبة جداً مع ال app اللي بتشتغل real-time.
ال " " " " " " " streaming.
ال " " " مع ال SPA.

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

Q) ايه الفرق بين الـ node js والـ browser؟

Browser	NodeJS
بيعمل run كود الـ js جوا الـ html بيعمل في الـ web page .	بيعمل run كود الـ js ناحية السيرفر بيعمل بالجهاز خارج الـ browser
بيوفر V8 engine وبعدها web API للتعامل مع الـ dom .	بيوفر V8 engine و core modules للتعامل مع الـ server والـ OS
مبيقدرش يتعامل مع الـ file sys	بيخلي الـ js يتعامل مع الـ file sys
لازم احط الـ js في html عشان يشتغل	اقدر اعمل stand alone app بالـ js لوحده .

Q) هل كل أكواد الـ js نقدر نشغلها في الـ nodejs؟

لا ، ده بيعتمد على الأكواد اللي الـ nodejs بيعملها ، يعني الحاجات اللي جاية مع الـ V8 engine

Q) ايه هو الـ node js REPL؟

هو عبارة عن الـ interactive cli الخاص بالـ node ، اللي بيسمحلنا نكتب ونشغل كود كل في السطر الواحد في المرة الواحدة .

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

Q) ايه هو الـ REPL ؟

* معظم لغات البرمجة بتوفرلنا الـ REPL وده عبارة عن tool لما بيسمحلنا نجرب ونشغل أكواد اللغة دي من الـ terminal.

* الـ REPL إختصار لـ Read-Evaluate-Print-loop

يعني : R ← اقراء كود من الـ user.
E ← افهم اللي بيكتبه وشغله.
P ← اطبعلي الناتج.
L ← ارجع اقراء منه كود تاني.

```
> 6 + 5
11
>
```

طالما بيشغل جملة

Q) هل الـ node لغة GUI ؟ لا، بيشتغل كـ tool فقط.

Q) ازاي نشغل الـ REPL ؟ اكتب أمر node فقط.

```
>> node ↵
```

Q) ازاي نشغل ملف js ؟ ↵ في الملف

```
>> node
>> node index.js ↵
```

Q) ازاي نقراء الـ command line argument اللي الـ user بيدخله في الـ node ؟

. كل الـ arguments بتتخزن في array اسمه argv في الـ process object.

(ex) console.log(process.argv[2])

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

(2) Node js Architecture :-

١- ايه هو الـ thread ؟ ايه انواعه ؟
الـ thread هو المسار اللى الـ code بيـ execute فيه.
يعنى هو مسار تشغيل الكود، بينقسم لنوعين :
① single thread.
② multi thread.

٢- ايه الفرق بين الـ single / multi thread ؟

الـ single thread هو مسار واحد لكل الـ tasks اللى فى البرنامج، يعنى البرنامج هيشتغل بشكل متسلسل.
* ميزته : مبيستهلكش resources كتير من الجهاز، وسهل جدا نتوقع مسار الكود.
* عيبته : بطىء، لانه بيعمل blocking للكود.

الـ multi thread هو إن إحنا نعمل مسار تشغيل لكل task لوحده، يعنى هنشغلهم بشكل متوازى مع بعض فى نفس الوقت.
* ميزته : بيسرع عملية التشغيل.
* عيبته : بيستهلك resources كتير من الجهاز.

٣- الـ blocking : هو التنفيذ الطبيعى للكود بترتيبه يعنى line by line بحيث الكود مش هيتنفذ إلا لما اللى قبله يخلص.
الـ non blocking : هو إن إحنا مبنعطلش مسار التنفيذ، بحيث الكود اللى بياخد وقت مش هيعطل اللى بعده.

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

@) ايه الـ lifecycle بتاعة الـ node؟
@) ازاي الـ node بتشتغل؟

١- الـ node بتستخدم الـ event-driven model يعني بتستخدم الـ events عشان تتبع تنفيذ الـ functions.

٢- الـ lifecycle بتاعة الـ node :-
١- اولاً الـ client هيبعت request لـ webServer.
٢- الـ node هيستقبل كل الـ requests اللي جايه ويحطهم أو هيعملهم emit في الـ event queue (هو queue بالتالي اول request دخل هو اول واحد هيخرج)

٣- الـ requests اللي في الـ event queue هتتنفذ واحد واحد في الـ event loop.

٤- الـ event loop هيقسم الـ events اللي جايه لقسمين،
← قسم بيعمل blocking، يعني بيحتاج resources خارجية أو بمعنى تاني بياخد وقت عشان ما يتنفذ، والنوع ده من الـ request بيتشال من مسار التنفيذ وبيتحط في الـ thread pool عشان يتنفذ هناك (الـ thread pool عبارة عن مجموعة thread بيتنفذوا بشكل parallel).

← القسم التاني مبيعملش blocking بالتالي الـ event loop بيرجع الـ response بتاعة للـ client اللي باعته.

← الـ thread اللي بيخلص في الـ thread pool بينبه الـ event loop، بعد كده الـ event loop هياخدها ويرجعها للـ client بتاعه.

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

Q) هل الـ node js هو single thread أم multi thread؟

هي single thread يعني من single threaded event loop
ولكن متبهلش blocking للكود من خلال إنها بتستخدم
الـ event loop والـ thread pool عشان تهندل الـ
asynchronouse code.

Q) ازاي الـ node js بتهندل الـ concurrency؟

الـ concurrency يعني التزامن، يعني أكتر من task بتنفذ
في نفس الوقت، الـ node js بيهندل الـ concurrency
باستخدام الـ event loop والـ callback system بتاع الـ node js
، بالتالي نضمن الـ responces اللي راجعة من الـ thread pool
تتنفذ واحد ورا واحد، مش واحد مع واحد.

Q) إيه هو الـ callback؟

هي function بتتبعت كـ argument لـ function تانية
وبتتنفذ بعد ما الـ function اللي اتبعتت فيه تخلص الأول.

الغرض من الـ callback إن إحنا نرتب تنفيذ الـ async code

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

③ modules , npm

Q) ايه هو الـ module؟

هو ملف كل نقدر ننقله reuse في ملف كل تاني من غير ما نعيد كتابة الأكواد / الـ logic من أول وجديد .

Q) ايه الغرض من الـ module؟

الغرض منه هو الـ reusability وتقسيم مشروع الـ كل لـ modules وكل module بيختص بـ logic معين .

Q) ازاي ننقل module ونستخدمه؟

أولاً : هننقل export للـ logic اللي عايزين نشاركه في ملف كل تاني .

ثانياً : هننقل import الـ module ، وهنستقبل الـ logic اللي راجعلي منه في variable .

```
① let x = () => {      };
  module.export = {x};

② let y = require('module الـ مسار');
  y.x();
```

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

Q) ايه انواع الـ modules اللى هنتعامل معاها؟

٣ انواع:

① core module : ودية الـ built-in modules اللى جاية مع الـ node.

② 3rd party module : ودية الـ modules الخارجية اللى بنعمل install.

③ custom module : ودية الـ module اللى بنعمل بنفسنا.

Q) ايه هو الـ package manager؟

مبدائياً الـ package هو الـ library او الـ framework، هو عموماً عبارة عن أكواد جاهزة للغة معينة.

كل لغة برمجة بيكون ليها الـ package manager tool بتاعها بحيث أي حد يقدر يعمل package باللغة دية ويرفعها في الـ pm الخاص بيها عشان باقي الـ developers يعرفوا يوصلولها.

الـ npm هي package manager tool بتساعدنا في إدارة الـ modules اللى بننزلها في الـ node project.

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

@ ايه هو الـ package.json ؟

* ده ملف بيحتوي على الـ metadata بتاعة الـ node project ، الملف ده بيعتبر قلب المشروع اللي بيحتوي على اسامي كل الـ modules اللي إحنا نزلناها في المشروع.

* فيه نوع رابع من انواع الـ modules اسمه ، @ native module وده module مكتوب بلغة غير الـ JS زي الـ C++ مثلاً.

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

(4) Node JS webServer

مصطلحات هامة

١- الـ client هو اى برنامج بيبعت طلب للسيرفر ،
او اى برنامج بيطلب خدمة من السيرفر.

٢- الـ server هو برنامج بيقدم خدمة معينة ، نوع السيرفر
بيعتمد على نوع الخدمة اللى بيقدمها. عندنا
مثلاً : web server ، mail server ، file server.

٣- الـ protocol هو طريقة التواصل بين الـ client والـ server
نوع الـ protocol بيحدد نوع الـ server .

٤- الـ IP هو عنوان الجهاز اللى بيشغل الـ server
فى الـ network .

٥- الـ port هو المكان اللى بيشغل السيرفر فى الجهاز ،
هو عامل زى اللوضة اللى بتحط فيها البرنامج.
اى جهاز computer بيوفر جواه 65,535 ports .

٦- الـ DNS هو اسم unique بيخص IP معين ، الغرض منه
هو تسهيل اسم السيرفر للـ end-users .

٧- الـ http هو الـ protocol المسئول عن التواصل
بين الـ client والـ webServer .

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

Q) ازاى نعمل webServer بال node؟

بنستخدم core module تبع ال node js اسمه 'http'، وده بيمكننا نعمل webServer.

```
1- const http = require('http');
2- const hostIP = '127.0.0.1';
   const port = 3333;

3- const server = http.createServer();
   server.on("request", (req, res) => {
       res.write('hello');
       res.end();
   });
4- server.listen(port, hostIP);
```

Q) ازاى نقرا المعلومات اللى جاية من ال request؟

```
let url = req.url;
let segment = req.url.split('/');
let method = req.method;
```

Q) ازاى نعمل routing؟ ١- هنقرا ال endpoint من ال url
٢- هنستخدم switch statement.

```
switch(url){
    case '/about' : =
                    break;
}
```


الممسوحة ضوئيا بـ CamScanner

7 ② ازاى نبعت response للـ client؟

```
8 res.writeHead(200, {'content-Type': 'text/plain'});
  res.end('hello');
9
  res.setHeader('content-type', 'text/html');
10 res.end('<h1>hello</h1>');

11 res.setHeader('content-type', 'text/css');
  res.end('h1{color:red}');
12
  res.setHeader('content-type', 'image/jpg');
1 let favIconBuffer = fs.readFileSync('./1.jpg');
  res.end(favIconBuffer);
2
  res.setHeader('content-type', 'application/json');
3 res.end(JSON.stringify( ));
```

4 ملحوظة هامة

5 ① روح إقرأ من الـ documents عشان تتعرف تفاصيل أكثر عن الـ http module.

6 ② الـ fs module هي core module بنستخدمها للتعامل مع الـ file system.

7

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

Express.js Session (2)

1- intro & project setup.
2- handle GET / POST Requests (routing).
3- Middleware.
4- route parameters.
5- routers.
6- query parameters.
7- cookies and sessions.
8- sending files
9- dynamic content & Template engine
10- express generator

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

① intro :-

Q) ايه هو ال expressjs ؟

ال express.js هو web framework لل node.js بيسهل علينا بناء server-side app والتعامل معاه .

ال express.js نقدر نبني بيه MVC web app كامل ، ونقدر نبني بيه REST APIs .

ال express.js بيسهلنا نهندل ال http req وال http res ، ونقدر نعمل routes حسب ال url وال http method ، ونقدر نعمل middleware functions .

Q) امتى نستخدم ال express ؟

لما نعوز نبني web app أو APIs بال node.js ، وده لان ال express بيسهل علينا بناء ال server-side app ،

Q) هل فيه structure معين بتمشي عليه في ال express ؟

لا خالص ، ال express هو unopinionated framework يعني مش بيجبرك بـ structure معين ، انما هو flexible اقدر اقسمه زي ما انا عايز .

على عكس ال Angular مثلا هو opinionated جدا ، يعني لازم امشي على structure معين وانا رحتعامل بيه .

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

@ ازاى نعمل setup لـ express project؟

١-
```
> mkdir project1 ↵
> cd project1 ↵
> npm init -y ↵
> npm i express ↵
> npm i -D nodemon ↵
> code . ↵
```

٢- هنعمل directory "src" نحط فيه ملف الـ "index.js" وده هيكون الـ entry point للمشروع.

٣- في ملف الـ package.json هنجهز الـ scripts اللى هنـ run بيها.

```
"scripts": {
    "start": "node ./src/index.js",
    "dev": "nodemon ./src/index.js"
}
```

الـ nodemon الغرض منه إنه يراقب التغيير اللى بيحصل في الكود/الملف، ويعمل restart للسيرفر.

٤- في ملف الـ index.js:

```
const express = require('express');
const app = express();
const port = 3000;
app.listen(port, () => console.log(`express server run on port: ${port}`));
```


https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

Friday
26
March

الجمعة
١٧ برمهات ١٧٣٧ ق

٢٦
مارس
١٣ شعبان ١٤٤٢ هـ

شرح الكود اللي فات ؟

أول سطر هنعمل import للـ express module ، مع ملاحظة إن الـ express module بيعمل export لـ express function بالتالي نقدر نعمل invoke لـ express.

الـ express function بيعمل return لـ instance من express app اللي هنستخدمه عشان نعمل السيرفر / الـ application.

ثانياً عملنا instance من الـ express app ، بالتالي نقدر نبدأ ونبني الـ server-side app.

ثالثاً عشان نبدأ نستقبل الـ requests من الـ client ، لازم نعمل listen على الـ port اللي هنقوم فيه السيرفر.

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

② handle requests in express (routing)

Q) ايه هو الـ get request؟

هو عبارة عن request method بنستخدمه مع الـ http protocol عشان نحدد للسيرفر إن إحنا عايزين ناخد resource موجودة عنده ،
الـ resource ممكن يكون أي حاجة نقدر ناخدها من السيرفر ، هي في الآخر عبارة عن data / information موجودة في السيرفر.

الخلاصة: الـ get request هي طريقة للوصول لـ information موجود عالسيرفر.

Q) ازاي نعمل implement لـ get request في الـ express؟

أولاً ، الـ express app instance بيحتوي على method جاهزة لكل request method ، وكل method جاهز منهم بيستقبل 2-parameters :-

الأول: هو الـ route path اللي هيتطابق مع الـ url ، ودة عبارة عن المكان بتاع الـ resource في السيرفر.

الثاني: هو الـ callback function اللي هتستقبل بيه الـ request ، ودة بيستقبل 2-parameters ، وهما الـ request object والـ response object.

(ex) app.get("/home", (req, res) => { });

path " — handler "

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

(ملحوظة)

(١) الـ request object بيحتوى على كل المعلومات اللى تخص الـ request اللى الـ client بعتة.

(٢) الـ response object بنهندل بية كل البيانات اللى تخص الـ http response اللى هترجعة/هنبعتة للـ client.

(٣) الـ resource المقصود بية المعلومات اللى هنبعتل للـ client، ممكن تكون أى data هنقدر نبعتل.

Q) ايه الفرق بين الـ post request والـ get request؟

الـ get بنستخدمة عشان ناخد resource من الـ server.
الـ post " " نحط resource فى الـ server.

- الـ resource يعنى الـ information أو الـ data.
- فى الـ post request إحنا بنبعت الـ resource فى الـ body بتاع الـ request.
- الـ request object بيحتوى على property اسمة "body" وده بيحتوى على الـ resource اللى فى الـ http request body.

```
(ex) app.post("/createResource", (req, res) => {
        console.log(req.body);
        res.send(201);
     });
     app.use(express.json());
```

لازم نعرف السيرفر ازاى يحول parse للـ data اللى هتتبعت، وده عن طريق استخدام middleware.

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

③ Middleware

@) اية هو ال middleware؟

- هي مجرد function عادية بيتنفذ invoke أثناء ال lifecycle بتاع ال request. حتى إن ال request handler اللي بنبعته كـ callback يعتبر middleware بدليل إنه بيستقبل "next" parameter.

- ال next parameter هي function بتستدعيها invoke ال middleware اللي بعدي، ال next middleware.

- أي middleware function بيقدر يعمل access لـ 3-parameters وهما: ال request object، ال response object، ال next function، بالتالي إحنا في ال middleware نقدر نعدل ال request اللي جاي، ونقدر نعدل ال response اللي هيتبعت، ونقدر ننفذ ال invoke لل middleware function اللي بعدي بالتالي نقدر ننفذ chain middleware.

(ex) app.get('/home', (req, res, next) => { });

ال request handler يعتبر middleware لأنه:
① بيتنفذ invoke أثناء ال request lifecycle.
② بيعمل access لل 3-parameters الأساسيين لل middleware.

ال parameter بعد ال route path يعتبر middleware.
app.get('/', (req, res, n) => {}, (req, res, n) => {}, ...);

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

(ex)

```
app.get(
  "/books",
  (req, res, next) => {
      console.log('before handling request');
      next();
  },
  (req, res, next) => {
    // response
      res.send('hello');
      next();
  },
  (req, res, next) => {
      console.log('finished executing the get request');
      next();
  },
  () => {
      console.log('the end');
  }
);
```

دلوقتي ممكن نعمل global middleware ونحطه في كل ال route اللي عندنا، وال middleware دي هتتعمل invoke ما بين مرحلة إستقبال ال route، ومرحلة ال request handling.

'/books', middleware, request handler

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

Q) ازاى نعمل apply لـ middleware ؟

بنستخدم الـ use() method من الـ express app instance
وبنحط فيه الـ middleware اللى عايزينه يـ run مع
كل request بيتبعت للـ server .

Q) ايه انواع الـ middleware اللى هنتعامل معاهم ؟

① built-in middleware وده تبع الـ express نفسه .
② custom middleware وده بنعمله بنفسنا .

(ex)
```
app.use(express.json());
app.use((req, res, next)=>{
    console.log(req.url);
    next();
});
```

الـ middleware functions بيتعملهم invoke
الـ request بيوصل للسيرفر ، وقبل ما الـ route
بتاعه يتعمله run .

http request /home → app.use( ) → app.get('/home', ()=>{})

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

(4) route parameter :- (request.params)

@) ايه هو الـ route parameter ؟

* هي dynamic value بتتحط من segment معين في الـ url وبتعرفه من الـ route باستخدام colon قبل اسم الـ parameter .

ex) https://localhost:3000/books/13

```
app.get('/books/:item', (req, res) => {
    console.log(req.params.item);
});
```

* الـ route parameters بيتحطوا في الـ params object على شكل key-value pairs .

* القيمة اللي بتتحط في الـ route parameters ممكن تكون رقم أو string عادي .

* الـ route الواحد نقدر نحط فيه أكثر من route parameter

* الغرض من الـ route parameter هو إن إحنا نبعت data للـ webserver .

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

(5) routers

Q) ايه هو الـ router وليه بنحتاجه؟

الـ router باختصار عبارة عن مجموعة من الـ routes اللي ليهم علاقة ببعض.

الغرض من الـ router في الـ express هو:

١- نشغل separation of concerney بحيث نقدر نفصل الـ routes في modules لوحدهم بعيداً عن الـ express app.

٢- نحسن الـ structure بتاع المشروع
١- نحسن الـ readability والـ maintainability.

Q) ازاي نشغل express router؟

هنستخدم الـ Routers method بتع الـ express module
الـ Router method بيرجع router object نقدر نشغل بيه routes.

احسن structure هو:
- نشغل folder نجمع فيه كل الـ router modules.
- نجمع كل الـ routes اللي ليهم علاقة ببعض في router.
- كل router هنشغله في js file / module لوحده.

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

(ex)

src/routes/book.router.js

```
const {Router} = require('express');
const router = Router();

router.get('/', );
router.post('/', );
router.put('/:bookId', );
module.exports = router;
```

~~exp~~

src/index.js

```
const express = require('express');
const bookRouter = require('./routes/book.router.js');
const app = express();

app.use(express.json());
app.use('/api/books', bookRouter);

app.listen(3000, () => { });
```

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

⑥ Query parameter (request.query)

Q) ايه هو ال query parameters؟

هي مجموعة parameters على شكل Key-value بيكونوا query string، ال query parameters بيتحطوا في نهاية ال url بعد علامة ال(?) وبنفصل بين كل parameter والتاني بـ (&).

ال query string هو الكلام اللي بعد ال(?) في نهاية ال url.
ال query parameter هو ال Key-value pair جوا ال query string

الغرض من ال query parameter هو إن إحنا نبعت data لل webServer

Q) ايه الفرق بين ال route parameter وال query parameter؟
Q) امتى نستخدم route parameter أو query parameter؟

بنستخدم ال route parameter لما نحب نعرف main resource معين في السيرفر زي مثلاً نحب ال product اللي ال id بتاعه بـ 1 ← https://localhost:3000/products/1

بنستخدم ال query parameter لما نحب نبعت data نشغل بيها search أو filter أو pagination أو sorting أو نشغل manipulate على response. https://localhost:3000?x=15

```
app.get('', (req, res) => console.log(req.query.x));
```


الممسوحة ضوئيا بـ CamScanner
https://www.linkedin.com/in/mohamadamir/

(7) cookies and sessions

Q) ايه هو ال cookies وليه هو مهم؟

* ال cookies هي شوية data السيرفر بيخزنها في ال client-side في مكان تبع ال browser اسمه cookies وال data دي بتتبعت مع كل request يروح للسيرفر بالتالي السيرفر هيفتكر ال client.

* الغرض من ال cookies ان إحنا نخزن data عايزين السيرفر يفتكرها، وده لأن ال http protocol هو stateless يعني كل request بيعتبره "كأنه جديد تماما"، بالتالي ال cookies ظهرت عشان نخلي ال http request يحتوي على state.

Q) ازاي نعمل cookie ونحدد ال expire بتاعها؟

```
response.cookie('visited', true, {maxAge: 10000});
```

ال options أو ال configuration بتاعة ال cookie.

ال ١st ال cookie ويكون string في قيمة ال cookie وممكن تكون string أو object.

```
>npm i cookie-parser
const cookieparser = require('cookie-parser');
app.use(cookieparser());
request.cookies
```


https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

Q) ايه هو الـ sessions؟

- الـ session بتوية data تخص client معين، بتخزن في السيرفر عشان السيرفر يفتكر معلومات معينة عن الـ client ده في كل مرة يبعت request.

- الـ session بينقسم لـ session data و session id
- كل session data بيكون ليه unique id،
الـ session data بتتخزن في الـ server-side وبتبعت الـ sessionid في الـ cookie عشان يتخزن في الـ client-side ويتبعت مع كل http request.

Q) ايه وجه التشابه والاختلاف بين الـ session والـ cookie؟

الاتنين عبارة عن data عايزين السيرفر يفتكرها، والاتنين الغرض منهم إن احنا نضيف state للـ http request.

الاختلاف في المكان اللي بنخزن فيه الـ data،
في حالة الـ session بنخزن الـ data في الـ server-side
في حالة الـ cookie ـــــ ـــــ في الـ client-side

بيفضل نستخدم الـ session مع الـ sensetive data.

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

Q) ازاى نستخدم ال session من ال express ؟

```
1- npm i express-session
2- const session = require('express-session');
3- app.use(session({
      secret: "ab17c15 99 cc8folmierz",
      resave: false,
      saveUninitialized: false,
   });
```

options دية
أو configuration
خاصة بال session

```
4- request.session.userInfo = {username: '', age: ''};
```

كده إحنا بنعمل modify لل session
modify بنعمل لل session
ال session بيتم إرسال cookie data (فيه ال session id)
لل client-side

```
5. const {userInfo} = request.session;
```

أو

```
request.session.userInfo
```

https://www.linkedin.com/in/mohamadamir/

الممسوحة ضوئيا بـ CamScanner

(8) send files in express :

* عشان نبعت file من ال response بنستخدم method تبع ال response object إسمها sendFile() وبتستقبل منا ال absolute path بتاع ال file من ال system .

* بيفضل لما نحط ال absolute path بتاع أي file ، يفضل تخلية OS agnostic يعني بيقدر يعدل شكل ال path حسب ال OS اللي ال server شغال عليه .

* فيه built in middleware تبع express اسمه express.static() ، ودة middleware بيرجع ال static files اللي ال user طالبها من ال request من غير ما نعملهم read ونحطهم في ال response بنفسنا ، وكمان بتسهلنا نفصل الملفات ال static في directory لوحدهم ونقدر نوصلهم بسهولة جدا في ال html عن طريق إن إحنا نحط (/) فقط ، ودة كده هيشاور على ال folder اللي شايل ال static files .

static files زي ال css وال js وال media files .

* عشان نخلي ال path يبقى OS agnostic بنستخدم ال path module .

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

```
1- const path = require('path');

2- app.use(express.static(path.join(__dirname, 'static')));

3- const homepath = path.join(__dirname, './views/home.html')
   const imgpath = path.join(__dirname, './imgs/1.jpg');

   response.sendFile(homepath);
```

هنحط ال absolute path بتاع الملف اللى عايزين نبعته مع ال response .

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

⑨ Template engine :-

* ال template engine هو software بيساعدنا نوصل لل underlying data ونعمل logics معينة جوا كود ال html ، يعني بيمكننا نكتب كود backend مع كود html في نفس الملف وفي النهاية بيطلع للـ end-user ملف html pure .

* اشهر انواع ال template engines في ال express.js هما :
① EJS
② Jade / pug
③ hbs

اسهلهم وابسطهم كـ syntax هو ال EJS .

---

⑩ ازاى نستخدم ال ejs template ؟

```
1- >npm i ejs ↵
2- app.set('view engine', 'ejs');
   app.set('views', 'pages');
```

لـ اسم الـ folder اللي حايط ملفات ال html .

```
3- app.get('/', (req, res) => {
     res.render('index');
   });
```

→ اسم ملف ال ejs اللي هنعمله render وتبعته في ال response كملف html .

```
4- pages/~~index.html~~ → pages/index.ejs
```


https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

ازاي نبعت data لل ejs file ؟

```
response.render('home', { name : "hello" });
```

ايه انواع ال tags في ال ejs template ؟

١- interpolation tag ← <%= %>
← بنعوض فيه ال js expression .

٢- scriptlet tags ← <% %>
← بنكتب فيه js code زي ال loops
وال conditions وال function call .

٣- comment tags ← <%# %>
← ممكن نعمل بيه disable لكود معين .

٤- include tags ← <% include ملف %>
← نقدر نعمل 'reuse' للملف ejs تاني .

```
<%= name %>
<% if( ){ %>
<% }else{ %>
<% } %>
```

<% include('مسار ملف ال ejs') %>

<%# %>

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

(10) express generator

* الـ express generator هو الـ tool ال الخاص بالـ express.js ، بنستخدمه عشان يبني الـ basic structure والـ boilerplate بتاعة الـ express app .

* ليه نستخدمه؟ لانه بـ command line واحد هيعمل generate لـ express application كامل بالـ boilerplate والـ configurations والـ structure اللي نقدر نعتبره كـ start point للمشروع .

Q) ازاي نستخدم الـ express generator؟

1- >npm i -g express-generator ↵
2 >express --help ↵

2- >express -e newApp ↵ (-e هيعمل ejs template)

3- >cd newApp ↵
>npm i ↵
5 >npm run start ↵

6 الخلاصة هي إن الـ express generator عبارة عن tool بيبنالنا الـ basic ideal structure اللي نقدر نبدأ بيه أي express application .

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

* شرح الـ structure اللى بينتجه الـ express generator

bin	بيحتوى على الـ www وده عبارة عن الـ server root directory أو الـ entry point اللى هيحتوى الـ compiled code.
public	بيحتوى على الـ assets / الـ static files.
routes	بيحتوى على الـ custom modules اللى بنعملها وكل الـ router modules والـ controllers
views	بيحتوى على الـ template files اللى هما الـ UI code ← jade ، pug ، ejs.
app.js	ده الـ main file ويعتبر قلب الـ application اللى هنعمل فيه import للـ modules والـ routers والـ middlewares والإعدادات بتاعة الـ app عموماً
package.json	بيحتوى على الـ metadata لكل الـ packages اللى هنحتاجها.

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

Session (3)

1- connecting to MongoDB
2- Authentication & Authorization & JWT
3- express validator
4- REST API
5- Upload image in Nodejs

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

① connect to MongoDB

للتعامل مع أي database بنحتاج حاجتين :

① database server : ده ال software اللي هيقوم ال db وبيه execute ال query language .

② database client : ده ال software اللي هنا connect بيه بال db server وهنبعت منه ال queries

db server زي مثلاً : mongoDB .

db client زي مثلاً : mongoshell ، RoboT3 ، mongoose .

٢- ال mongoose هو library js بتمكننا نعمل connection بال mongoDB server ونتعامل معاه من ال Node.js .

٣- ال mongoose بيشتغل بـ concept ال ODM / ORM وده عبارة عن عملية تمثيل ال db table / ال db collection على هيئة object بالتالي نقدر نتعامل مع ال db كـ oop من غير ما نكتب أي query language بنفسنا .

٤- ال MongoDB URI ده عبارة عن ال link اللي هنا connect بيه بال MongoDB server .

mongodb://localhost:27017/bl7

- mongodb: ال protocol الخاص بالـ mongo
- localhost: ال host اللي عليه ال mongo
- 27017: ال port اللي بيه run فيه
- bl7: اسم ال db اللي إحنا عايزينها

الممسوحة ضوئيا بـ CamScanner

@ إزاي ن connect الـ MongoDB مع الـ Node باستخدام الـ mongoose ؟

أولاً لازم نشغل الـ mongoDB server .
ثانياً لازم نجهز الـ db اللي هنتعامل عليها .

```
>npm i mongoose ↵
```

```
const mongoose = require('mongoose');
const dbURI = 'mongodb://localhost:27017/blz';
mongoose.connect(dbURI),
.then(()=>console.log('MongoDB connected'))
.catch((err)=>{
    console.log('MongoDB connection error:', err);
    process.exit(0);
});
```

كل collection / table عندك في الـ db هتتمثل
بـ schema و model ، الـ model دة هو الـ object
اللي هتتعامل بيه CRUD على الـ collection اللي بيمثلها .

يفضل كل model تتعمل في module لوحدها .

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

(ex) models / User.model.js

```
const mongoose = require('mongoose');
const Schema = mongoose.Schema;
const personSchema = new Schema({
    name: String,
    age: Number,
    email: String
});
const Person = mongoose.model('Person', personSchema);
module.export = Person;
```

5- استخدم الـ model عشان تعمل الـ CRUD :

create new document :

```
const Person = require('./models/User.model.js');
const p1 = new Person({
    name: 'Hero',
    age: 16,
    email: 'example@gmail.com'
});
p1.save().then(() => console.log('success'))
        .catch((err) => console.log('fail', err));
```

Read all documents

```
Person.find()
.then( data => console.log('all persons', data))
.catch( err =>      );
```

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

Read by id

```
const personId = '167811';
Person.findById(personId)
.then( data => )
.catch( err => );
```

Update document

```
const personId = '11819FS';
const newdata = {age: 30};
Person.findByIdAndUpdate(personId, newdata)
.then( () => )
.catch( err => );
```

Delete document

```
const personId = '11819FS';

Person.findByIdAndDelete(personId)
.then( () => )
.catch( err => );
```

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

② Authentication :-

الـ Authentication بيتكلم عن فكرة انت مين ؟
الـ Authorization ~ ~ ~ تتقدر تعمل ايه ؟

١- العمليات الاساسية في الـ Authentication هما :
① login
② register
② logout

٢- الـ auth credentials هي الـ data اللي بنـ identify بيها الـ user زي الـ email والـ password .

٣- الـ OAuth هو عملية الـ authentication من خلال 3rd party application زي google ، facebook ، ... .

٤- ايه الفرق بين الـ encryption والـ hashing ؟

← الاثنين عبارة عن عمليات cryptography / تشفير
الغرض منهم هو حماية الـ sensetive data بتاعة الـ users .

← كل واحد منهم هو algorithm مختلف لتشفير الـ data .
← في الـ encryption algorithm نتقدر نشفر الـ data ، ونتقدر نرجع نفك التشفير ، يعني هي 2-way function .
اما
← الـ hash هي oneway function يعني مقدرش افك التشفير .

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

ازاى نعمل hashing لل password

```
1- npm i bcrypt
2- const bcrypt = require('bcrypt');

3- bcrypt.hash(mypassword, 10, (err, hashed)=>{
        if(err){                    salt
        }else{
        newpass = hashed
        }
   });

4- bcrypt.compare(pass, hashedpass, (err, isMatch)=>{
        if(err){ }
        else if(isMatch){ }
        else{ }
   });
```

ال salt هى random string بنضيفها على ال password
قبل ما نعمل hash ، وده عشان نصعب عملية تخمين
ال password من ال hash .

ال bcrypt بيرجع promise object بالتالى نقدر نستخدم
async/await مثلا ، عشان يشكل الكود يبقى انظف .

لازم نشفر ال password بتاع ال user قبل
ما نحطه فى ال db .

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

أشهر طرق انشاء ال auth sys في ال express
هما:
① authentication with passport.js
② authentication with jwt

ال token هي عبارة عن بيانات تسجيل ال user لكن متشفرة، بنستخدمه عشان نتأكد هل ال client اللي بعت ال request هو user عمل login ولا لأ.

ال token لازم نبعته مع كل request عشان نعرف ال server إن إحنا مسجلين منه.

Q) ازاي نعمل generate و verify لـ JSON web Token؟

```
1- npm i jsonwebtoken
2- const jwt = require('jsonwebtoken');
   const payload = {userId: 1236};
   const secretKey = 'حط الرقم اللي انت عايزه';

3- const token = jwt.sign(payload, secretKey, {expiresIn: '1h'})

4- jwt.verify(token, secrectKey, (err, decoded) => {
       if(err) {  }
       else {  }
   });
```

ال data اللي في ال token

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

③ express validator (data Validation)

* دي عبارة library بتوفرلنا مجموعة middlewares
نقدر نستخدمها عشان نـ validate الـ data الي
جاية للسيرفر.

1- > npm i express-validator ↵

2- const {check, validationResult} = require('express-validator')

← الـ check ← هو middleware function بتبدأ اسم
الحاجة الي عايزين نـ validate عليها في
الـ request وبتحدد نوع الـ validation.

3- router.post('/', [
    check('name').notEmpty(),
    check('pass').notEmpty(),
  ],
  (req, res) => {
    const error = validationResult(req);
    if(error.isEmpty()){ }
    else{ }
  });

اقرا الـ documentation

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

④ RESTFull API

ايه هو الـ Restfull API؟

هي طريقة لنقل الـ data بين الـ applications على شكل Json format من خلال الـ http.

ايه الشروط بتاعة Restful architecture؟
ايه الـ constraints بتاعة Restful؟ الـ Rest API:

١- لازم نفصل بين backend والـ frontend تماماً.
٢- لازم الـ API مخزنش أي حاجة عن الـ client اللي جايلة.
٣- لازم نحدد شكل معين لنقل الـ data (غالباً بيكون Json)
٤- الـ response اللي جاي من الـ Rest API ممكن نكيشه في الـ browser.
٥- الـ Rest API مش لازم يكون الـ layer الوحيد بين الـ client والـ server، يعني الـ API يقدر يستخدم API تانية.
٦- الـ Rest API مبيهتمش ايه نوع الـ client / البرنامج اللي بيعمل الـ request.

اي Rest API هيتكون من:
① url endpoint واحد لكل الـ http methods.
② هنضيف route parameter id للـ update والـ delete.
③ كل enpoint هيمثل manage لـ resourse معين.

(ex) products/		get	Client ⟷
		post	
	:id	update	
	:id	delete	

endpoint = http method + path = POST / product

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

our API

/products

get | post

الـ endpoint عبارة عن: httpmethod + path
(ex): GET /products

/products/{id}

get | put | delete

(ex)

```
app.get('/products', (req, res, next) => {
    // bussiness logic
    res.status(200).json({ });
});
```

ليه هو الـ CORS؟ إيه هو الـ CORS error؟

الـ CORS إختصار لـ cross origin resourse sharing
يعني بنقل، share الـ data بتاعتنا على أكثر من origin
، الـ origin هو الـ domain بتاعي.

الـ CORS error بيظهر لو حاولنا نعمل share / data
الـ resource لـ webserver على domain غير الـ domain
بتاع الموقع، يعني الـ server والـ app على domain مختلف.

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

ايه هو ال CORS policy؟

ده policy بيقول مينفعش domain يعمل request لـ domain تاني، لازم يبقى من نفس ال domain، الغرض من ال CORS هو ال security بحيث نمنع أي request يجي من application مش تبع ال server.

ازاي نسمح لـ origin معين يبعت request؟
ازاي نلغي ال CORS policy من عاك origin معين؟

← هنحدد في ال header ليه ال origin / ال domain اللي مسموحلة يبعت request منك.
← طالما هنعترض ال request يبقى على طول هنعمل middleware.

```
app.use((req, res, next)=>{
    res.header('Access-Control-Allow-Origin', '*');
    res.header('-----Methods', '*');
    res.header('-----Headers', '*');
    next();
});
```

أي domain

* كده احنا حددنا للسيرفر:

١) ايه ال domains / origins اللي مسموحلها تبعتلة request؟
٢) ايه ال http methods اللي مسموح تتبعت؟
٣) ايه نوع ال header اللي مسموح نستقبله؟

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

⑤ Upload Image :- multer

- عندنا ٣ طرق نرفع بيها الـ uploads :
  1- نخزن الـ file عندنا في السيرفر .
  2- نخزن الـ file في الـ database على شكل Base64 .
  3- نخزن الـ file في cloud server زي firebase مثلاً .
- افضل واسرع طريقة هو ان إحنا نخزن الـ files مباشرة عندنا في السيرفر .

- Multer هو middleware بيسهلنا نرفع الـ multipart/form-data تحديداً بنستخدمه عشان نرفع الـ uploading files .

الـ Multer بيرفع الـ multipart form فقط

1- > npm i multer

2- images/ هنجهز directory نرفع فيه الصور :

3-
```
const multer = require('multer');
const path = require('path');
```

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

4.
```
const storage = multer.diskStorage({
  destination: (req, file, cb) => {   callback
    cb(null, path.join(__dirname, './images'));
  },
  filename: (req, file, cb) => {
    const uniquePrefix = Date.now() + '-' + Math.round(Math.random*1E9);
    cb(null, uniquePrefix + '-' + file.originalname);
  },
});
```

5.
```
const upload = multer({storage: storage});
```

6.
```
app.post('/', upload.single('image'), (req, res) => {
    (يجيب ال form field اللى جوا ال file.
  console.log(req.file);
  res.status(200).json({ });
});
```

7 إزاى نعمل get لل image؟

* هنستخدم ال express.static() ونديله ال path بتاع ال static folder.

* ال express.static() دى middleware function.

```
app.use(express.static(path.join(__dirname, './images')));
```

(ex) http://localhost/(images folder ال file بتاع ال)

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

NodeJS

في هذا الكورس سوف اشرح مفاهيم اكثر من شرح مفاهيم بدون تطبيق عملي.

مفاهيم => 50% استيعاب.
مفاهيم + تطبيق => 100% استيعاب.

https://www.linkedin.com/in/mohamadamir/

الممسوحة ضوئيا بـ CamScanner

week ①

1. introduction to backend
2. introduction to nodeJS
3. node lifecycle
4. synch Vs Asynch
5. course syllabus
6. js refresh

https://www.linkedin.com/in/mohamadamir/

CamScanner الممسوحة ضوئيا بـ

١- ايه هو ال website ؟

هو عبارة عن مجموعة ال files أو ال documents اللي بنقدر نوصل من خلال شبكة ال internet .

٢- هل ال frontend كفاية لوحده ؟

لا طبعاً لأن ال frontend لوحده بيعملنا شغل static application فقط ، يعني application مبيحتويش على shared data .

٣- ايه هو ال backend ؟

هو software بيعملنا شغل shared data ، أو هو ال software المسؤول عن ادارة البيانات الخاصة بال end user وتخزينها بشكل دائم في ال server .

٤- ايه هي مدارس انشاء ال backend ؟

مدرسة ال SSR وده اختصار لـ server-side rendering وده معناه إن ال backend هو المسؤول عن بناء وتعبئة صفحات ال html بال content قبل ما يبعتها للـ client .

مدرسة ال CSR وده اختصار لـ client-side rendering وده معناه إن ال client-side هو المسؤول عن تعبئة صفحات ال html بال content بالتالي وظيفة ال backend مختلفة عن ال frontend .

https://www.linkedin.com/in/mohamadamir/

الممسوحة ضوئيا بـ CamScanner

NO.:
Date:

٥- ايه هو الـ server ؟

ما هو إلا جهاز كمبيوتر شغال متوصلة من أي جهاز كمبيوتر تاني ومن أي وقت احنا عايزينه من خلال الـ network سواء كانت local network أو global network.

٦- ايه هي انواع الـ server ؟

① dedicated server for each service :

← ودي عبارة عن إن إحنا بنقسم الـ product أو بنقسم الـ services اللي بنشغلها على أكثر من server، بحيث كل خدمة هتشغل على server.

← النوع دي بنستخدمه في المشاريع الضخمة.

② single dedicated server handle all tasks.

← هنحط كل الـ services اللي هنشغلها في server واحد.

← النوع دي بنستخدمه في المشاريع الصغيرة.

انهي أفضل نوع فيهم؟!

اول نوع اللي هو dedicated server for each service ودي لأن لو فيه service وقعت أو server وقع، دة هيوقفش باقي الـ services التانية.

بالتالي ظهر concept في الـ backend اسمه الـ microservice وهو ان اجزاء كل جزء موجود في الـ application لـ service منفصلة لوحدها، بحيث كده احنا كل الـ services دي تكلم بعض ودي هتحسن الـ performance وبسهل عملية التعديل والصيانة.

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

٧- ايه هو الـ node.js ؟

الـ node مش لغة برمجة ومش framework، ولكن الـ node هي runtime environment js بتسمحلنا نبني server application باستخدام الـ js.

٨- ليه نستخدم nodejs ؟

١- لأن باستخدام الـ nodejs هنستعمل لغة برمجة واحدة من الـ frontend والـ backend.
٢- لأن الـ nodejs بيوفر asynchronous feature يعني أكتر من task بيشتغل في نفس الوقت من غير blocking.
٣- لأن الـ nodejs بيوفر الـ sockets وده بنعمل بيه real-time applications زي الـ chats.
٤- لأن الـ nodejs بيشتغل كل حاجة فيه في single thread لأن الـ node application أصلاً هو lightweight ضعيف
٥- لأن الـ nodejs بيدعم الـ microservices architecture، يعني نقدر نعمل بيه microservices architecture.
٦- لأن الـ nodejs نقدر نعمل بيه Rest APIs.
٧- لأن الـ nodejs ~ ~ ~ chat bots.
٨- لأن الـ nodejs بيستخدم الـ event-driven يعني فيه حدث بيحصل بالتالي الـ node بيرد عليه، يعني الحدث بيحدث.
٩- الـ event-driven والـ non-blocking الفكرتين دول خلوا الـ nodejs تبقى خفيفة lightweight.

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

Date :

9- ايه الفرق بين ال asynchronouse وال synchronous ؟

① ال synch بيـ run ال code خطوة خطوة بالترتيب ولو فيه process بياخد وقت بيستناه لغاية ما يخلص بالتالي ال synch بيعمل blocking لل execution .

② ال asynch بيركن ال process اللي بتاخد وقت على جنب لغاية ما تخلص بعد كده يكمله run من غير ما يعمل blocking لباقي الكود .

هلاقي شرح تفصيلي عن ازاي ال node اتعملت وازاي بتشتغل في الاجندة بعنوان :
(learn and understand NodeJS)

10- امتى بيعمل blocking ؟ امتى بيعمل non-blocking ؟ امتى نقول إن الكود عمل blocking ؟

* بيعمل blocking لو لاقينا ال thread عطلان على حاجة

* بيعمل non-blocking لو عندنا multi-threads او حتى عندنا single-thread ولكن بطريقة ما بنقدر نشغله كأنه multi-thread بالتالي طول ما انا ببعت requests ال single thread بيفضل يشتغل ومش بيعطل ومن نفس الوقت ماشي في process تانية كانت مطلوبة منك .

* لو ال thread يشتغل على process ويقدر يكمل process تاني معاه فده كده non-blocking ، لكن لو ال thread عطل فده blocking .
وقف
استنى

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

course syllabus

1- js refresh
2- node.js basic
3- express.js
4- Routes, models, middleware
5- MVC
6- node with MySQL
7- ORM sequelize
8- MongoDB
9- ODM mongoose

10- validation (JOI)
11- Authentication
12- Authorization
13- Async Requests
14- REST API
15- Relation two ways using mongoose

16- Error handling
17- file uploads
18- pagination
19- sending Email
20- social media login

21- CRN Jobs & Create PDF & QR code

22- deployment
23- template engines
24- sessions
25- socket.io
26- GraphQL
27- Redis DB
28- Docker
29- NestJS

https://www.linkedin.com/in/mohamadamir/

CamScanner الممسوحة ضوئيا بـ

js refresh

١- الـ variable هو مكان في الميموري بنشيل فيه data معينة.

٢- الـ js هي loosly type يعني بنحضر الـ type بتاعها بسهولة، لأن نوع الـ variable بيحدد نوع الـ value اللي متخزن فيه.

٣- الـ primitive datatype هي الـ datatype الأساسية اللي اللغة بتوفرها، وبتكون عبارة عن قيمة واحدة متغيرة.

٤- الـ non-primitive datatype هي نقدر نشكلها من الـ primitive، يعني هي عبارة عن مجموعة من الـ primitive datatype.

٥- الـ variable هو مكان في الـ memory بنـ hold فيه data معينة والـ data دي ممكن تتغير.

٦- الـ syntax هو طريقة الكتابة.

٧- طريقة إنشاء الـ variable بتنقسم لجزئين:
* الـ declaration وهو إن أحجز مكان في الميموري وأديله اسم.
* الـ assignment وهو إن أحط قيمة في المكان ده.

٨- الـ scope هي الحدود اللي نقدر نـ access الـ variable فيها.

٩- الـ function هي reusable block of code بشكل بسيط.

١٠- الـ plus operator بيشتغل في حالتين:
← لو الـ two operands كانوا أرقام، الـ + هيشتغل كجمع.
← لو أحد الـ operand كونوا string، الـ + هيشتغل concat

https://www.linkedin.com/in/mohamadamir/

الممسوحة ضوئيا بـ CamScanner

١١ أي operator ثاني غير الـ(+) بيعمل implicitly casting لـ number.

1 + 1 + 1 + '1' → '31' type conversion to string

1 + 1 + 1 + +'1' → 4 ~ ~ ~ number

1 + 1 + 1 + -'1' → 2 ~ ~ ~ number

2 + '2' → 22

2 * '2' → 4

---

١٢ الـ variable اللي متعرفة بـ var Keyword بيبقى scope global وال scope بتاع الـ file كله ماعدا في حالة الـ function.

اي حاجة متعرفة جوا الـ function بتكون local scope.

١٣ الـ JS فيه concept اسمه hoisting يعني رفع، يعني الـ JS اثناء الـ runtime بتعمل رفع للـ var declaration اعلى الـ scope بتاعها وتديلها قيمة اولية بـ undefined، وجزء الـ assignment بيفضل في مكانه.

```
console.log(x)
var x = 5;
```

→

```
var x = undefined;
console.log(x);
x = 5;
```

الـ hoisting بيخلينا نعرف نـ access الـ variable قبل ما يتعرف.

١٤ من مشاكل الـ JS هي hoisting، والـ scope في الـ JS الـ function هي local scope

الـ if ~ global scope

الـ for loop ~ global scope

المشكلة دي اتحلت لما الـ let ظهرت في الـ ES6، الـ let بتخلي المتغير يبقى local وال scope اللي هو متعرف فيه بس.

https://www.linkedin.com/in/mohamadamir/

الممسوحة ضوئيا بـ CamScanner

١٥ الـ let حلت مشكلة الـ var .

* المتغير اللي متعرف بـ var بيكون global scope على مستوى الـ file ماعدا لو اتعرف جوا function وبيحصلله hoisting .

* المتغير اللي متعرف بـ let بيكون local scope اللي هو متعرف فيه فقط وميحصلش فيه hoisting. يعني بيكون local scope على قد الـ scope بتاعه .

١٦ الـ hoisting هو عملية رفع الـ variable declaration من أعلى الـ scope اللي هو فيه مع ترك الـ variable assignment في مكانه .

١٧- الـ try catch بنستخدمهم عشان نهندل بيهم أي error بنشأ في الـ code / أي error محتمل يحصل في الـ code عشان ميوقفش الـ execution .

ازاي نستخدمه؟
في الكود اللي إحتمال يحصل فيه مشكلة / error هنحطه جوا الـ block بتاع الـ try ، ولما يحصل error هيتبعت كـ argument لـ catch بعدين الـ catch هي run .

١٨- الـ destructing هي عملية تفريغ محتويات الـ object أو محتويات الـ array في variables بحيث كل قيمة هتفرغ في variable خاص بيها .

(ex) let x = [1, 2, 3];
let [x, y, z] = x;

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

١٩- ال this Keyword جوا ال method بتشاور على ال object اللي
بيحتوي ال method دية .

* ال this جوا method جوا object بتشاور على ال object .

* ال this جوا function جوا method جوا object بتشاور
على ال window object ، ودي لأن ال function اللي جوا method
بتكريت this خاصة بيها ولا تعود لل object انما تعود
على ال global window object .

* ال this جوا arrow function جوا method جوا object
بتشاور على ال object ، ودي حصل لأن ال arrow function
ملهاش this يعني مبتكريتش this خاص بيها زي ال
function العادية بالتالي بتاخد ال this بتاعة ال parent
بتاعها .

٢٠- الفرق بين ال arrow function وال function العادية هي
إن ال arrow function معندهاش this خاص بيها
انما بتاخد ال this بتاعة بباها .

٢١- ال arrow function معندهاش this .
ال arrow function بتاخد ال this بتاعة بباها .

٢٢- بنحتاج ال try catch عشان نمنع إن ال error يوقف
الكود ، يوقف ال server

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

٢٢- ال try catch هو concept بنهندل بيه ال error عشان ميوقفش الكود.
الكود اللي احنا شايفين انه ممكن يعمل error هنحطه في ال try، بحيث ال try هتجربه لو اشتغل تمام ولو عمل error ال catch هتمسكه وباقي الكود هيكمل من غير ما يوقف.

٢٣- اي error في ال backend هيوقع السيرفر على عكس ال frontend بالتالي في ال backend لازم نهندل اي error ممكن يحصل.

٢٤- ال this في جوا ال js لازم بتتلون، يعني قيمتها بتختلف حسب الأماكن اللي هي فيها، بتتلون على لون المكان اللي هي فيه.

٢٥- ال this جوا function جوا method جوا object قيمتها بتكون ال window وليس ال object نفسه، وده لأن ال function بتعمل لنفسها this جديدة خاصة بيها. الحل؟
عندنا حلين:-
١) نخزن ال this بتاع ال method في variable ونستخدم ال variable دي في ال function بدل ال this.

٢) نستخدم arrow function بدل من ال function العادية.

٢٦- ال "use strict" هو directive بيجبرك تكتب كود js سليم من غير اخطاء أو bug أو error.

ال directive كلمة هو أي كود بيوفر توجيه أو تعليمات لغرض محدد

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

١٧ الـ js بتشتغل non-blocking يعني الكود اللي هياخد وقت في التنفيذ إركنه بيشتغل لوحده على جنب لغاية ما يخلص من غير ما توقف باقي الـ code.

١٨ الكود الـ synchronous بيـ run بالترتيب بالتالي احنا عارفين الكود هيبدأ امتى وهينتهي أمتى.

الكود الـ asynchronous بيـ run حسب اللي يخلص تنفيذ الأول وليس بالترتيب بالتالي مبنعرفش مين هيتنفذ قبل مين.

١٩ الـ concept الـ callback في الـ js ظهر عشان يهندل الـ async code.
بنستخدم الـ callback عشان نرتب تنفيذ أكواد الـ asynch، عشان نعمل blocking، عشان نخلي أكواد الـ asynch تتنفذ كـ synch بالترتيب.

٢٠ الـ promise ظهر كبديل للـ callback عشان يحل مشكلة الـ callback hell بحيث طريقة الكتابة بقت أبسط وأسهل.

٢١ عندنا ٣ طرق نهندل بيها الـ asynchronous code.
① نستخدم الـ callback function يعني نبعت الـ function كـ parameter في function تاني.
② نخلي يعمل return لـ promise.
③ نستخدم الـ sugary syntax اللي اسمه async, await.

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

٣٣- ال promise لما نعمل منه instance بيستقبل function كـ initial value لل instance ، وال function دي بتستقبل 2 parameters وهما resolved وده اللي بتطلع من ال then() ، و rejected ~ ~ ~ ~ من ال catch().

٣٢- لما بنعمل ال function تعمل return لـ promise فده آلية دخلت ال function في اشارة مرور يعني أول ما يدخل الاشارة حتى مش run أي حاجة إلا لما نندة على resolved().

```
function test(){
    return new Promise(resolved, rejected) =>{
        resolved();
        rejected();
}  })

test.then(resolved).catch(rejected)
```

٣١- الفرق الرئيسي بين ال primitive وال non-primitive هي ال value اللي بيستقبلها.

← ال primitive بيستقبل value ثابتة.

← ال non-primitive بيستقبل عنوان ال value من ال memory. ال value بتاع ال object وال array بيكون reference وال reference بيروح لـ value.

https://www.linkedin.com/in/mohamadamir/

الممسوحة ضوئيا بـ CamScanner

٢٥- • الـ program هو ملف فيه أكواد / تعليمات للـ CPU .
• الـ process هو program متحولة execution في الـ CPU .
• الـ thread هي الوحدة الأساسية للـ execution في الـ process ، كل process لازم يحتوي على thread واحد على الأقل .
• الـ thread هو الـ core أو المسار اللي بنبعت فيه الأكواد للـ CPU عشان يشغلها .

٢٦- single threaded يعني الـ process بيقدر يشغل task واحد في المرة الواحدة .

multi threaded يعني الـ process بيقدر يشغل أكثر من task في نفس الوقت مع العلم إن كل thread بيشغل task واحد فقط

٢٧- إزاي الـ node / الـ js هي single thread وبتقدر تشغل أكثر من task في نفس الوقت ؟

لو بسبب الـ architecture بتاع الـ nodejs
عندنا الـ V8 engine هو single thread ولكن الـ program اللي مدمج معاه هو اللي بيسمحلنا ننفذ asynch tasks .
المنفذ المساعد + المنفذ الأساسي
في حالة الـ browser بيكون V8 + web API
في حالة الـ server بيكون V8 + libUV
أو V8 + node API

سواء الـ web API أو الـ libuv فالاثنين بيشتغلوا أكواد الـ js مع الـ V8 engine في نفس الوقت ، مع العلم إن الـ V8 engine هو المسار الرئيسي للتنفيذ الفعلي لكود الـ js .

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

١- ال js هي loosly datatype يعني بتفرق ال type بتاعها بسهولة بالتالي ال variable بيقبل أي type عادي.

(ex)

```
var x = 5;
x = 'hello';
x = true;
```

٢- ال scope هو المكان اللي بتشاف فيه المتغير.

٣- ملف ال js كله عبارة عن object يعني أي حاجة بنكتبها جواه تعتبر key-value جوا ال object دي.

٤- ال variable اللي متعرف بـ var keyword بيكون global scope على ال file كله ما عدا داخل ال function body.

index.js

```
var x = 5
```

→

index.js

```
{ var x = 5
}
```

٥- أي variable متعرف بـ var بيتعمله hoisting يعني بيتم رفع ال declaration بتاع ال variable من أعلى ال scope اللي هو فيه قبل ما يحصل execution للكود يعني قبل ما ينفذ الكود.

Name Tel. Email

يونيه ٢٠١٦

السبت	٤	١١	١٨	٢٥
الأحد	٥	١٢	١٩	٢٦
الاثنين	٦	١٣	٢٠	٢٧
الثلاثاء	٧	١٤	٢١	٢٨
الأربعاء	١ ٨	١٥	٢٢	٢٩
الخميس	٢ ٩	١٦	٢٣	٣٠

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

٧- هيتم الـ var Keyword:
١) بتكون دايماً global في مستوى الـ file كل ماعدا لو اتعرفت جوا function.

٢) بتعمل hoisting.
٣) بتقدر تناديها قبل الـ declaration بتاعها.
٤) تقدر تعمل re-declare يعني تقدر تعمل variable تاني بنفس الاسم في نفس الـ scope.

٨- علشان الـ hoisting بتحصل قبل مرحلة التنفيذ/الـ execution.

```
console.log(X)
for(var X = 0; X < 6; X++){
}
console.log(X)
```

o/p
undefined
6

٩- الـ var بتبقى global scope في أي block ماعدا الـ function.

```
console.log(X);
function run(){
  var X = 5;
}
run();
console.log(X);
```

o/p
error : X is not defined

١٠- الـ var بتعمل hoisting وبتاخد قيمة مبدائية بـ undefined

الـ let والـ const بتعملهم hoisting ولكن مبياخدوش قيمة مبدائية.

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

```
console.log(x);
var x = 5;
```

→

```
var x = undefined;
console.log(x);
x = 5;
```

```
console.log(x);
let x = 5;
```

→

```
let x;
console.log(x)
x=5;
```

∴ o/p = error, can't access x before initialization

ايه ميزة ال let عن ال var؟

١) بتبقى local scope بال block اللى هى فية على عكس ال var اللى دايماً بتبقى global او اى حاجة ماعدا ال function.

٢) بتعمل hoisting ولكن مبتعملش initialization بـ undefined زى ال var بالتالى مقدرش نعمل access قبل ما يتعمل initialization.

٣) ميقدرش نعمل re declare على نفس ال scope.

٤) يقدر نعمل re-assign.

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

هل تقدر تعمل reassign ال const لـ i ؟

```
for( let i=0; i<5 ; i++){
    console.log(i);
}
```

o/p
0
1
2
3
4
5

```
for(const i=0; i<5 ; i++){
    console.log(i);
}
```

o/p
0
error

ازاى نعمل for loop بال const وتشتغل من غير error ؟

الحل إن إحنا نعمل loop على array ، لأن ال const مينفعش نغير قيمته لو قيمته كانت primitive value .

ايه الفرق بين ال primitive datatype وال non-؟

ال primitive datatype هى ال data اللى بتخزن فى ال memory/ ال variable بشكل مباشر ، يعنى بتعمل pass by value .

تذكر إن ال variable هو مكان فى ال memory بيشيل قيمة data .

ال non primitive هى ال data اللى بتخزن فى ال variable بشكل غير مباشر ، يعنى بتشيل فى مكان وتخزن عنوان المكان دة فى ال variable ودة بتسمية " pass by reference "

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

ازاي نفرق بين ال data ال primitive وال non?

لو ال variable بتاعها قيمة تفضل
بيعرف القيمة بتاعة primitive .
انما
لو بتحول عنوان القيمة بتاعة اذا فهي non-primitive .

نقدر نعمل reassign لل nonprimitive لكن إحنا
مش بنغير ال reference ولكن بنغير القيمة تفضل .

```
const x = 5;
x = 14;
```
error

```
const x = [1, 2];
x[0] = 3;
```
قيمة ال x متغيرتش
ولكن القيمة اللي ال x
بيشاور عليها اتغيرت

pass by value ← بتستقبل ال value تفضل
في ال variable .

pass by reference ← بتستقبل عنوان ال value وليس
ال value نفسها .

القيم ال non-primitive بتتخزن في ال Heap .

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

١٥- اية انواع الـ arguments ؟

(١) positional argument دى الـ argument اللى الترتيب / الـ order بيفرق فيه .

```
(ex) function sub(x, y){
        return x - y
     }

     console.log( sub(5,10)); // -5
     console.log( sub (10, 5)); // 5
```

(٢) named argument دى الـ argument اللى بنبعته وبنحدد له اسم parameter يتخزن فيه .

```
(ex) function sub({x , y}){
        return x - y;
     }

     console.log(sub({x:10 , y:5})); // 5
     console.log(sub({y:5 , x:10})); // 5
```

٢٠- ازاى تعمل named argument فى الـ كلاس ؟

بنبعت الـ object لـ function كـ parameter وبنعمل destruction .

٢١- الـ arrow function ملهاش this بالتالى بتاخد الـ this بتاعة بابها .

الـ function العادية بتعمل this خاصة بيها .

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

week (2)

npm Vs yarn.
how to start nodejs application.
modules type in nodejs.
create server using "http" module.
nodemon
url & routing
http lifecycle & methods
, how to send data in the request?
buffer
middleware

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

١- ايه الفرق بين الـ yarn Vs الـ npm؟

١) الاتنين عبارة عن package manager للـ JS.
٢) الـ npm بينزل مع الجهاز by default مع الـ nodejs، ولكن الـ yarn بننزله باستخدام الـ npm.

٣) الـ yarn عملته شركة facebook عشان تحل مشكلة بطء الـ npm.
٤) الـ yarn اسرع من الـ npm:
٥) ميزة الـ yarn انه لو بنزل package والنت قطع، لما النت يرجع الـ yarn هيكمل تحميل من عند آخر نقطة وصلها، في حين إن الـ npm هيرجع يحمل من أول وجديد.

٢- ايه هو ملف الـ package.json؟

دة ملف بيحتوي على المعلومات الاساسية الخاصة بالـ project والكل مشروع هنقدر نشغل المشروع من غيرها، زي مثلاً:
① الـ entry point الخاص بالـ application.
② الـ dependencies اللي الـ app هيحتاجها لما نرفعه.
③ الـ " اللي الـ app هيحتاجها اثناء مرحلة الـ development فقط.
④ الـ scripts / الـ commands اللي بنشغل بيها الـ application.

٣- npm init بيخلينا نملاء ملف الـ package.json بشكل يدوي.
npm init -y بينشاء ملف الـ package.json على طول.

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

٤- ايه هو الـ module وايه أهميته وانواعه؟

← هو ملف أو مجموعة ملفات بتحتوي على أكواد جاهزة مندقر نعيد استخدامهم في أي مشروع وفي أي وقت.

← الغرض من الـ module هو التنظيم والـ code reusability.

← انواع الـ modules ٣ أنواع في الـ nodejs :-

① core modules وديه الـ modules اللي نازلة مع الـ nodejs بتاعتي.
② " local وديه الـ " " بنكتبها بنفسنا.
③ " 3rd party وديه الـ " " طرف ثالث كاتبها وبننزلها باستخدام الـ npm.

٥- ازاي ننزل مشروع nodejs؟

- اولاً لازم ننشاء ملف الـ package.json ← npm init -y
- ثانياً هننشاء ملف الـ index.js / ملف الـ entry point.
- ثالثاً ننزل الـ nodemon وديه بتخلي أي تغيير في الكود يسمح في الـ server من غير ما نقفل السيرفر ونعيد تشغيله من تاني.
- رابعاً ننشاء / نشغل الـ server اللي هنكتب فيه كود الـ backend بتاعنا.

عندنا ٣ أنواع من الـ modules في الـ nodejs وهما:
١- core / built-in module وده اللي جاي مع الـ node.
٢- local / custom module وده اللي بنعمله بنفسي.
٣- 3rd-party module وده اللي بننزله بالـ npm.

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

٦- عشان نعمل create لـ server هنحتاج core module
إسمه "http".

الـ http هو object بيحتوي على method اسمها
createServer() بيستقبل callback function بتاخد
positional arguments (req, res, next).

```
const http = require('http');
http.createServer((req, res)=>{}).listen(3000);
```

السيرفر هو جهاز كمبيوتر كبير بنقدر نوصل لبرنامج شغال
عليه من خلال شبكة الانترنت أو الـ local network.

عشان نعمل backend لازم نشتغل على server، بالتالي
بشكل مبدأي هنحول الجهاز الحالي بتاعنا لـ server محلي
وده في مرحلة الـ development فقط لغاية ما نعمله deploy
على server حقيقي.

ليه بنسمي الـ entry point بـ index؟

← لأن أغلب السيرفرات في الـ configuration بتاعها
بتسمي الـ entry point file بـ index، بالتالي يفضل
تسميه index، يا إما تسميه بالاسم اللي
إنت عايزه ولكن تغير في الـ configuration.

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

٧- ال nodejs فيه كام نوع من ال module؟

فيه ٣ أنواع من ال module.
١) build-in module / أو core module اللي جاية مع ال node
٢) local module اللي بنعمله بنفسنا.
٣) 3rd party module اللي بننزلها بال npm.

ايه هو ال module أصلاً؟

هو ملف .js أقدر أعمل export للأكواد اللي فيه بحيث أقدر أعمل import / reuse ليوم ما أي ملف .js تاني، ال module هو reusable .js file

ال node.js عبارة عن modules / مبنية على ال modules، فكرة ال module ظهرت عشان أقدر أعمل structure لل .js files.

٨- لما نبدأ نعمل أي nodejs application لازم أول خطوة نعملها هي إنشاء ملف ال package.json لأنه هيحتوي على كل المعلومات الأساسية لتشغيل المشروع.

٩- ال node مبنية على ال modules.
ال express framework مبني على ال http module.
ال nest framework مبني على ال express.

ال node عبارة عن modules.
ال express عبارة عن middlewares.

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

10- `const http = require('http');`

`http.createServer((req, res, next)=>{`

ده المكان اللي هنكتب فيه الـ logic بتاع الـ backend

، ده المكان اللي هنكتب فيه الـ code (server/ backend)

، ده المكان اللي هيـ run كل ما نبعت request للسيرفر.

`}).listen(3000, ()=>{`

ده المكان اللي هيـ run مرة واحدة لما السيرفر يشتغل

، المكان ده ملوش أي لازمة غير إنه يعرفني إن السيرفر اشتغل / قام / شغّل

`});`

11- ليه هو الـ nodemon وازاي نشغله؟

الـ nodemon هو development tool بيـ watch أي متغير في الكود، لما يحصل بالتغيير بيقوم يوقف السيرفر ويعيد تشغيله من تاني بشكل تلقائي بدون تدخل مني.

ازاي نشغله:

① `>> nodemon index.js` ← الـ entry point

أو

② `>> nodemon` ←

لو عملنا run لأمر الـ nodemon هو تلقائياً هيروح على الـ package.json هيشوف اسم الـ entry point اية

https://www.linkedin.com/in/mohamadamir/

الممسوحة ضوئيا بـ CamScanner

NO.:
Date:
رقم:
التاريخ:

١١- ايه هو الـ url ؟

هو النمط أو هو اللينك اللي هنطلبه بين السيرفر، وبيتكون من:

① الـ protocol اللي السيرفر بيتكلم بيه، اللي من خلاله هنقدر نتواصل مع السيرفر.

② الـ IP address / الـ domain name وده عبارة عن عنوان السيرفر على شبكة الانترنت.

③ الـ port number وده رقم المكان اللي الـ application بتاعنا بيه run فيه على السيرفر.

④ endpoint وده اللي بنفرق بيه بين كل الـ urls، كل الـ urls بتاعة لازم يكون ليها endpoint مختلفة.

١٢- ايه هو الـ routing في الـ frontend والـ backend؟

الـ routing في الـ frontend هو التنقل بين الصفحات بناءً على الـ endpoint في الـ url.

الـ routing في الـ backend هو التنقل بين الـ services أو الـ logic أو الـ resources اللي السيرفر بيقدمها بناءً على الـ endpoint في الـ url.

الـ routing زي الـ conditions if بيتحكم في الـ flow بتاع السيرفر.

الـ router بيتكون من (http method + url)

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

ايه هي ال lifecycle بتاعة ال {http request}
ليه هي ال lifecycle بتاعة ال {client-server-architecture}

---

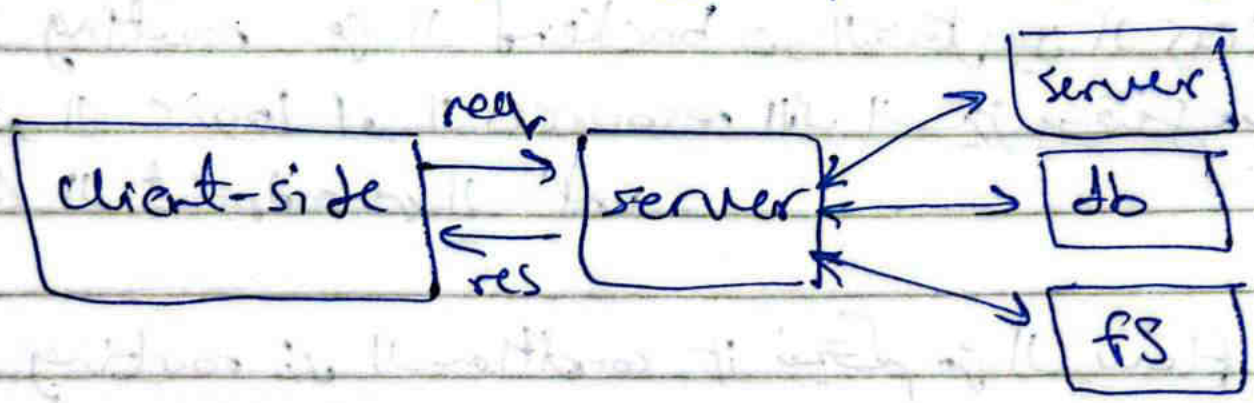
١) ال client بيبعت request لل server بيطلب منه خدمة / أو data معينة.

٢) ال server هيكلم service تانية أو هيكلم database أو هيكلم ال file system حسب ال requirement بتاعة ال request اللي اتبعتلة.

٣) ال server هيرد ب response يحمل ال data اللي ال client طالبة من ال request.

٤) كل request بيتبعت للسيرفر لازم يترد عليه ب response وإلا ال browser هيفضل يعمل loading لحد ما يحصل timeout.

٥) كل request بيكون ليه response واحد فقط، مينفعش ال request يترد عليه باكتر من response.



https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

١٥- ايه الفرق بين ال http وال https ؟

ال https هو النسخة ال secure من ال protocol http ،
(ال https = http + ssl)

• ssl يعني ال data اللي رايحة مابين السيرفر والـ client
متبعتش متشفرة .

• http بس من غير ssl يعني ال data اللي رايحة مابين السيرفر
والـ client هتتبعت زي ما هيا بدون تشفير .

• السيرفر اللي هتعمل deploy عليه لازم فيه support ال
https .

١٦- ايه الفرق بين ال MVC وال API ؟

في ال MVC السيرفر بيرد على ال client بـ html .
في ال API السيرفر بيرد على ال client بـ json .
الغرض من ال API هو ال software reusability .

ايه هو ال API timeout ؟ وامتى بيحصل ؟

هو الوقت اللي ال API بيستنى فيه السيرفر اية يرد عليه
ولو الوقت ده خلص من غير ما السيرفر يرد عليه ، فال API ده
هيعمل timeout . امتى بيحصل timedout ؟
① لو ال backend مبيرجعش response اصلاً .
② لو ال backend رجع response ولكن بعد ما ال timeout

١١- ايه هو الـ Http method؟

الـ Http method هي طريقة ارسال الـ data من الـ request ، وطريقة التعامل مع الـ data دية من السيرفر.

١٢- ايه الفرق بين الـ GET request والـ POST request؟

عندنا ٣ فروقات :

① أول فرق هو طريقة ارسال الـ data في الـ request .

← الـ get بيبعت الـ data في الـ url على شكل query params بالتالي الـ data بتبقى متشافة لل end-user.

← الـ post بيبعت الـ data في الـ request body بالتالي الـ data بتكون متشفرة بيخلي more secure .

② تاني فرق هو كيفية ارسال الـ request نفسه :

← الـ get request نقدر نبعته بسهولة من خلال الـ browser باستخدام الـ anchor tag / links أو الـ search bar.

← الـ browser ميقدرش يبعت إلا الـ get request فقط ، بالتالي عشان نبعت post أو put أو delete لازم نستخدم ajax js أو library js زي الـ axios ، fetch ، أو نستخدم الـ form tag.

③ تالت فرق هو طريقة التعامل مع الـ data في السيرفر :

← الـ get بيطلب resource من السيرفر.

← الـ post بيحط resource في السيرفر.

https://www.linkedin.com/in/mohamadamir/

الممسوحة ضوئيا بـ CamScanner

١٩- ازاي نبعت data من ال request ؟
اين الأماكن المتاحة لل end-user انه يبعت فيها data للسيرفر ؟
اين الأماكن المتاحة لل server انه يستقبل فيها ال data من ال request ؟

- عندنا ال request body ، عندنا ال headers ، عندنا ال params ، عندنا ال query .
- دول كلهم اماكن نقدر نبعت فيهم data من جوا ال request ، وكل واحد منهم عبارة عن object ، وكلهم بيرجعوا object
- يا إما فاضي يا إما بال data لو اتبعتت فيه .

٢٠- اين الفرق بين ال query وال param ؟

الاثنين بيظهروا في ال url بعد ال request ومكانهم في ال request url ، الفرق بينهم بيكون في طريقة الكتابة وطريقة الوصول ليها وهل هي optional ولا لا .

query	params
١) بيكتب كـ parameters في ال url http://baseURL/api?K1=V1&K2=V2 ...	١) بيكتب كـ path في ال url ex: http://baseURL/api/toto/1 app.get("/api/:x/:y", ...)
٢) بنستقبله باستخدام (req.query) ٣) ال query هو optional يعني ممكن نبعته وممكن لأ ، لأنه مبيظهرش في ال url pattern ، إنما بيتحط بعدين .	٢) بنستقبله باستخدام (req.params) ٣) ال params مش optional يعني لازم نبعت القيمة بتاعته ودي لأنه بيغير في ال pattern ال url .

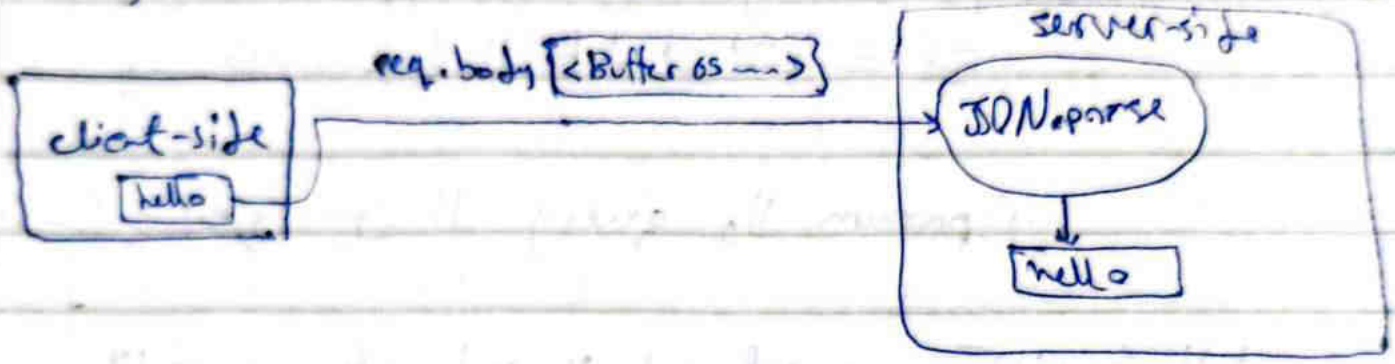
٢١- ليه قيمة ال query بتكون optional بينما ال param لازم نبعته ؟
لأن ال param بيغير في structure بتاع ال url إنما ال query بيتكتب بعد ال url .

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

٢٢- ال post request بيبعت ال data في ال request body
في buffer، يعني ال data اللي هنتعامل بال post هتتحول
لـ hexa code في buffer.

```
if(req.url== "/test" && req.method== "POST"){
    req.on('data', (chunk)=>{
        console.log(JSON.parse(chunk));
    });
}
```

لما بنعمل parsing لل buffer يعني
هنرجعه من hexa لـ string.



٢٣- ايه الحاجات اللي بنحتاجها واحنا بنعمل أي [API]

١) لازم نحدد ال router وده عبارة عن ال request url
وال request method.

٢) لازم نحدد ال data، هل هنبعت data في ال request ولا لا؟
ولو هنبعت data هنحطها فين في ال request؟ هل في ال body
ولا ال header، ولا ال params، ولا ال query.

٣) بعد ما حددنا ال router وال data، بعد كده هنبدأ نكتب
ال logic اللي ال API هيعمله.

٤) كل request لازم يرد عليه بـ response.

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

٢٤- ايه الفرق بين res.write و res.end و res.send ؟

اولاً الـ "res" هي الـ response object اللي بنستخدمها لإرسال response للـ client، الـ res.send، res.json، res.end، res.write بعمل كلهم حاجة لإرسال الـ response.
ايه الفرق ؟!!.

① الـ res.write():
- بنستخدمها عشان نبعت data للـ client.
- بتبعت الـ data كـ chunk يعني بتبعت الـ data كقطع، وممكن تبعت response لسة، هو بيكتب بس، بالتالي نقدر نبعت/نعمل أكتر من res.write() ورا بعض.

② الـ res.end():
- بتنهي الـ response process، يعني بتبعت الـ response للـ client.
- بتبعت الـ data اللي جواه مرة واحدة، وليس كقطع زي الـ res.write.

③ الـ res.send():
- خاصة بالـ express.
- بتساوي الـ res.write والـ res.end مع بعض مرة واحدة.
- بتقدر تبعت json أو string أو html أو buffer.

④ الـ res.json():
- هو زي الـ res.send() ولكن الفرق الوحيد إنه مبتبعتش html.
- وبتحول الـ object js لـ json.
- محول مخصوص عشان يبعت json data

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

- الـ middleware هي function بتـ run أثناء الـ lifecycle بتاعت الـ request.

- أي middleware في الـ nodejs هي عبارة عن function بتاخد (req, res, next) كـ positional argument والـ logic بتاعها يا إما { يرجع next() ويكمل اللي بعده } أو { يرجع error ويقف }.

- الـ next() لما يرن run، بيشغل الـ middleware اللي بعده على طول.

- الـ app.use() بتستقبل middleware أو أكثر وتحقنه في أي route مكتوب بعدها.

- لو مابين الـ app.use() يعترض route معين، يبقى هنكتب الـ path بتاع الـ route ده في الـ app.use().

```
const express = require("express");
const app = express();
const auth = (req, res, next) => {
    if (islogedIN) {... return next()}
    else { res.json({msg: "not user"}) }
}
```

أي middleware يرجع next أو error

```
app.get('/a', auth, ...);
app.get('/b', auth, ...);
```

(1) =

```
app.use(auth);
app.get('/a', ...)
app.get('/b', ...)
```

```
app.use('/a', auth);
app.get('/a', ...);
app.get('/b', ...);
```

(2)

https://www.linkedin.com/in/mohamadamir/

الممسوحة ضوئيا بـ CamScanner

١٦- الـ express.json() عبارة عن middleware بياخد الـ body
اللي جاي في الـ request ويعمله parse يعني يحول
الـ buffer لـ string .

١٧- الـ app.use() بيستخدمها نعمل middleware عين يعمل global
في أي route مكتوب بعد كود الـ app.use() .

١٨- الـ * بنستخدمه عشان نعمل default route .
app.get("*", ...);

١٩- خطوات التعامل مع أي module في الـ node :
① اعمله install .
② اعمله call / invoke .
③ ابدأ استخدمه

٢٠- أي request لازم يترد عليه بـ response واحد فقط
عشان كده أي function بتهندل الـ request لازم تعمل
return للـ response حيث أي حاجة بعد الـ return مش
هتـ run بالتالي متجنب أي error ممكن يحصل زي مثلا إني اهنا
بعت response تاني .

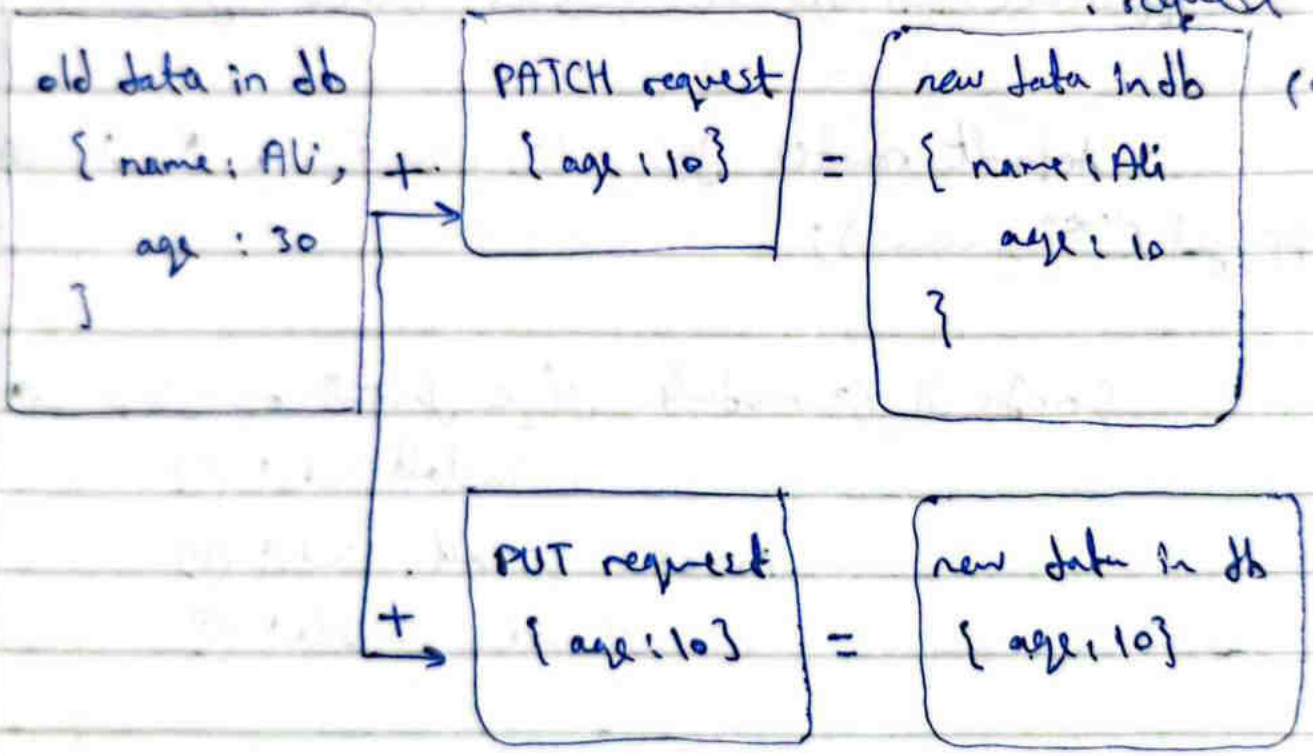
لو بعتنا أكثر من response ، أول response هو اللي هيتبعت
بعد كده هيحصل error لأن كل request بيكون ليه response
واحد فقط ، عشان كده دايما بنعمل نعمل return للـ response .

٢١- الفرق بين الـ PUT والـ PATCH إن :
الـ PUT بنستخدمها لما نعوز نعمل update للـ object كله على بعضه ،
الـ PATCH بيعمل update على جزء من الـ object .

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

@ ايه الفرق بين ال PUT method وال PATCH ؟

ال PATCH بيغير تغيير جزئي في ال object / ال data ،
ال PUT بيستبدل ال object كله بال data اللي مبعوتة في
ال request .



اذاً :

① استخدم ال PATCH req لو هتعدل field معين في ال data .
② استخدم ال PUT req لو هتعدل كل ال fields في ال data .

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

NO.:
Date :

week 3

1- how to send file using express
2- database intro
3- cors policy error
4- relations

https://www.linkedin.com/in/mohamadamir/

CamScanner الممسوحة ضوئيا بـ

١- ازاي نبعت file لل client ؟

index.html
index.js

at index.js
```
app.get("/", (req, res, next) => {
    res.sendFile(__dirname + '/index.html');
})
```

notes

١- عشان نبعت file بال node ، لازم نديله ال path بتاعه بالكامل ، يعني من أول ال root directory لغاية ال current (working) directory .

٢- ال nodejs لازم نبعتله ال path absolute وليس ال relative path .

٣- __dirname دي variable جاهز في ال node بيجيبلك ال absolute path لغاية ال ~~file~~ directory اللي إنت فيه .

٢-
```
const fs = require("fs");
const htmlContent = fs.readfileSynch("index.html");
app.get("/", (req, res) => {
    res.send(htmlContent);
    = res.end(htmlContent);
    = res.sendfile(__dirname + '/index.html');
});
```

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

٢- ايه الفرق بين: npm init و npm init -y ؟

• ال npm init بينشاء ملف index.js و package.json وبيخليني املاء بيانات ال package.json بايدي واحدة واحدة.

• ال npm init -y بيعمل inforce ال "yes" option بالتالي بينشاء ملف ال package.json على طول ببيانات default ولكن مش بينشاء ملف ال "index.js".

٣- اول خطوة لبدء أي مشروع بال node هو انشاء ملف ال package.json لان ده الملف اللي هيحتوي على المعلومات الاساسية لتشغيل المشروع.

٤- كل request لازم يترد عليه بـ response واحد فقط، بالتالي يفضل دايماً نعمل return للـ response لأن أي حاجة بعد ال return مش هتتنفذ بالتالي متجبش ال response تاني يتعبت عليه اللي بعدها.

https://www.linkedin.com/in/mohamadamir/

الممسوحة ضوئيا بـ CamScanner

ايه الطرق المختلفة لتخزين الـ data؟
ايه الفكره العامة اللي عايزين نوصل؟
ايه مشكلة تخزين الـ data في الـ array؟

الفكرة العامة هي إن احنا عايزين نحتفظ بالـ data في مكان ما نقدر نوصله ونعدل عليه.

1) أول طرق تخزين الـ data هي الـ "array" ولكن مشكلة انه بيخزن الـ data في الـ RAM بشكل مؤقت يعني بيخزن data volatile بالتالي البيانات مش هتتخزن بشكل دائم.

2) تاني طريقة هو إن احنا نخزن الـ data في file في الـ hard disk بحيث الـ data هتتخزن بشكل دائم (non-volatile) ولكن مشكلة الـ filebase:
1) صعوبة الوصول للـ file لأنه الـ path بتاعه هيختلف من OS للتاني.
2) صعوبة الوصول للـ data وصعوبة التعديل عليها.
3) الـ performance، مثلاً نحرك الـ file أو نكتب عليه هنحتاج في كل مرة نـ load الـ file.

3) ثالث طريقة لحفظ الـ data هي الـ database وهو عبارة عن file تخزن فيه الـ data بتنظيم معين. باستخدام الـ db نقدر نحدد شكل ونوع البيانات اللي هتتخزن ونقدر نحدد مين يقدر يوصل للـ data ومين لا، ويقدر يوصل للـ data ويعدل عليها بسهولة جداً.

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

٢- إيه اللي هنحتاجه عشان نشتغل بال db ؟

١) هنحتاج db server نشيل ونهندل فيه ال db .
٢) هنحتاج query language نتكلم بيها ال db .

٣- ال db server ممكن يكون local db server زي Xampp .
وممكن يكون cloud db server زي Mongodb Atlas .

٤- عيب ال cloud db server إنه بيحتاج نت عشان أقدر أوصله .
ميزته إني أقدر أوصله من أي مكان ومن أي جهاز وفيه شركة كبيرة
هي المسؤولة عن حماية البيانات بتاعتي والحفاظ عليها .

٥- عيب ال local db server إني مش هقدر أوصله إلا من خلال الجهاز
اللي متوصل بال local network اللي موجود فيه ال db server
، وأنا المسؤول عن حماية البيانات والحفاظ عليها .
ميزة ال local db server إنه مش بيحتاج نت عشان أقدر أوصله .

٦- الفكرة من ظهور ال db هو تسهيل الوصول للبيانات وتسهيل
التعامل معاها والتعديل عليها .

٧- ال SQL هي اللغة المسؤولة عن التعامل مع ال database .
اللغة اللي هنتشاور بيها ال db ونتكلم معاه بيها .
ال ERD هي رسمة بتوضح ال db structure بتاع المشروع
، بتوضح شكل ال db اللي هننفذه هيجي حاجة ازاي .

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

٩- عشان نشتغل db مع ال nodejs محتاج :

١) محتاج db server يشغل ال SQL code أي : Xampp

٢) محتاج 3rd party module لربط ال dbserver بال node app

أي : "mysql2" module .

١٠- ال developer هيتعاملوا مع ال db من خلال :

١) استخدام ال wizard الخاص بال dbms .

٢) استخدام ال SQL code .

٣) استخدام API بحيث هنشتغل ال db بحدود معينة .

١١- ال end-user هيتعامل مع ال db من خلال ال application .

١٢- ال wizard زي ال software tool أو ال interface اللي ال user يقدر من خلاله يشغل ال db باستخدام GUI وليس بال code .

CLI = SQL code

GUI = wizard

١٣- ايه هي مشكلة ال cors origin بتاعة ال resource sharing

دي مشكلة بتظهر لو حاولنا نعمل access / نتكلم ال server من خلال ال client / frontend والسيرفر مش سامح بدة

دي مشكلة بتحصل لأن ال backend مش سامح لأي حد انه يعمل access عليه بالتالي ال backend لازم يحدد مين يقدر يوصله ومين لا .

https://www.linkedin.com/in/mohamadamir/

الممسوحة ضوئيا بـ CamScanner

١٤. ايه هي ال db lifecycle ؟ ايه المراحل اللي بتمر خلال developer قبل ما يبتدي شغل وبعد ما يبتدي شغل ؟

١ مرحلة ال analysis :
← مرحلة تحليل ودراسة المشروع من وجهة نظر ال business وفهم متطلبات السوق .
← دية شغلانة ال product owner / system analyst
← بيطلع requirement doc أو user story .

٢ مرحلة ال database design :
← مرحلة تصميم الهيكل العام للـ db .
← تحويل ال req doc اللي ضرناه من ال product owner لـ ERD .
← دية شغلانة ال db designer .

٣ مرحلة ال database mapping :
← مرحلة تحويل ال ERD لـ schema بتوضح تفاصيل أكتر وليس شكل عام .
← دية شغلانة ال db designer .

٤ مرحلة ال db implementation :
← مرحلة بناء ال db / مرحلة تنفيذ ال db schema باستخدام ال DBMS المناسب .
← دية شغلانة ال db developer .

٥ مرحلة بناء ال UI / ال application (web-mobile-desktop)

٦ مرحلة ال client ودي تتفاعل ال end user مع ال db من خلال ال application .

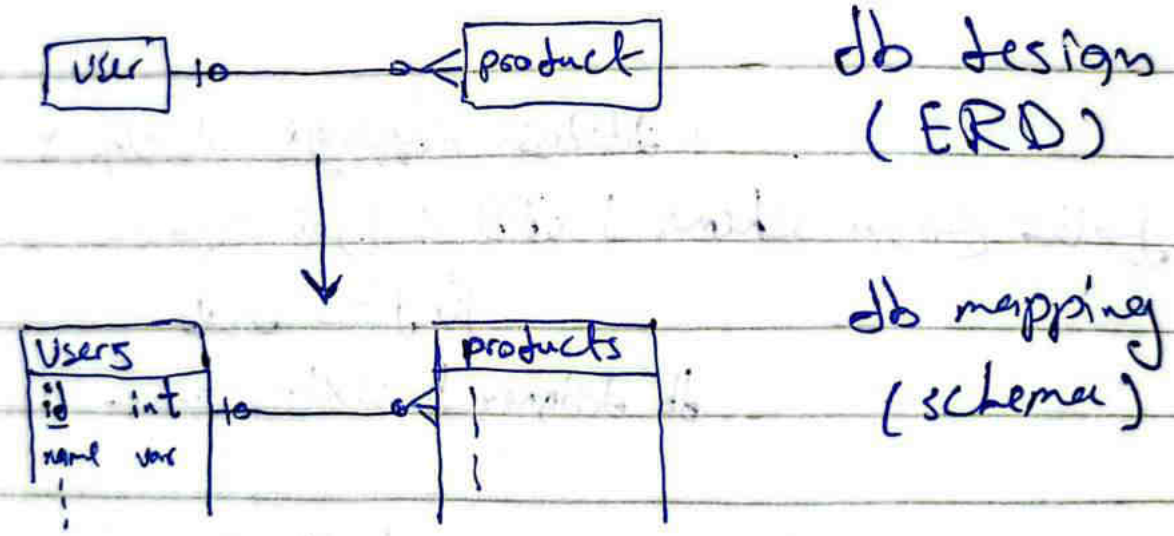
https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

ايه الفرق بين ال ERD وال schema ؟

ال ERD هي رسمة بتوضح الشكل العام لل db ، يعني
اسماء ال tables فقط والعلاقات بينهم .

ال schema هي ال ERD ولكن بتفاصيل يعني بتوضح
اسماء ال tables و البيانات اللي متخزن فيهم وانواع البيانات دي
والعلاقات .

ال db schema فيها معلومات اكتر بتستخدم في بناء/تنفيذ
ال db .



انا بشرح بتفصيل اكتر من ال database هتلاقيها
في اجندة ال PHP ووزق ال db بتاع ال iti .

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

16. مش هنقدر نبعت request لمش هنقدر نوصل
لـ resource معين من السيرفر من خلال الـ browser
إلا لما السيرفر يسمح بده، وإلا هيطلعلنا cors error

17. الـ backend لازم يحدد مين ليه صلاحية يقدر يوصله
ومين لأ. الـ مش موصلة هيتبقالك cors error

18. ازاي نحل مشكلة الـ cors origin؟
ازاي نحدد مين الـ clients اللي يقدر يوصل للـ resource اللي عندي؟

هنستخدم 3rd party module اسمه "cors".
لـ npm i cors ←
const cors = require("cors");
app.use(cors()) ← client/frontend كده انا بسمح لأي
انه يوصل للـ resources اللي عندي.

---

19. إيه تصنيفات أكواد الـ SQL؟

DDL ← بنستخدمها عشان نتعامل مع الـ structure بتاع الـ db أو الـ table
DML ← ~ ~ ~ مع الـ data نفسها.
DCL ← ~ ~ ~ مع التحكم في صلاحيات الوصول للـ db.
DQL ← ~ ~ ~ نجيب الـ data.
TCL ← ~ ~ ~ الـ transactions.

أكواد الـ DDL بتأثر على الـ structure نفسه وليس الـ data اللي جوا.
أكواد الـ DML بتأثر على الـ record/data اللي جوا الـ table.

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

عشان نشغل relationship في الـ ER لازم نتبع الخطوات التالية:

1- أول خطوة: حدد اسم للعلاقة. (يعني الجدول علاقته ايه بالجدول التاني)
يفضل اسم العلاقة يكون فعل.

2- ثاني خطوة: حدد درجة العلاقة، ودي بتحكم من عدد الجداول اللي العلاقة دي بتربط بينهم.
binary degree ← بتربط بين جدولين فقط.
" binary ← " " ٣ جداول أو أكثر.
" recursive ← " " الجدول ونفسه.

3- ثالث خطوة: حدد نوع العلاقة (الـ cardinality)
← علاقة one to one
← " OTM
← " OTM + MTO = MTM

4- رابع خطوة: حدد أهمية العلاقة (الـ participation)
هل الجدول ة/الـ record لازم يكون موجود optionally؟
لو لازم يبقى هو fully participation في العلاقة
لو optional " " partially في العلاقة

عشان نشغل relationship بين جدولين، لازم ولابد فيه جدول منهم يكون عايز حاجة من الجدول التاني.
الحاجة اللي هيكون بتكون الـ PK
والحاجة اللي هيأخد بيكون الـ FK.

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

٢) ال fk ما هو إلا ال PK بتاع table تاني .

٣) ال join هو عملية دمج جدولين ليهم علاقة ببعض وغرضهم .

٤) بنستخدم ال join عشان نجيب data من جدولين فيه relation ما بينهم ، فيه عامل مشترك بيربطهم ببعض .

٥- عشان نعمل relation بين جدولين ، لازم نعمل fk column في جدول يشاور على ال PK column في الجدول التاني .

أولاً نعمل fk column باستخدام ال wizard { Xampp

---

١) روح لل table اللي عايز تعمل fk column فيه .

٢) افتح تابة ال "Structure"

٣) افتح تابة ال "Relation view"

٤) حدد أن اسم ال constraint ويفضل الاسم يكون كالتالي :

FK_ اول حرف من اسم الجدول _ اول حرف من اسم الجدول

understand

التاني بحرف كبير ، الاولاني بحرف كبير

(الجدول اللي فيه ال fk) (الجدول اللي فيه ال PK)

٥) حدد تأثير ال ON DELETE ، يعني لو record اتمسح ، هل ال record اللي ليه علاقة بيه هيتمسح ولا لا .

٦) حدد ال column اللي انت عايزه يكون ال fk column .

٧) حدد ال table اللي عايزين نعمل relation معاه .

٨) حدد ال column اللي عايزين ال fk بتاعنا يشاور/يعمل reference عليه .

https://www.linkedin.com/in/mohamadamir/

الممسوحة ضوئيا بـ CamScanner

ازاي ~ ~ ~ بـ استخدام ال code ؟

relationship ~ ~ ~ ؟

```
create table `employee`
(id int,
 age int,
 name varchar(50),
 constraint `PK_ID` PRIMARY KEY (`id`)
 id int PRIMARY KEY
);

create table `customer`
( c_id int PRIMARY KEY,
  c_age int,
  c_name varchar(50),
  E_id,
  constraint `FK_E_C`
  foreign key (`E_id`)
  references `employee` (`id`)
  on delete cascade
  on update cascade
);
```

https://www.linkedin.com/in/mohamadamir/

CamScanner الممسوحة ضوئيا بـ

week (4)

folder structure in "ES5" Vs "ES6".
query Vs execute in "mysql2" module
ORM
sequalize

https://www.linkedin.com/in/mohamadamir/

CamScanner الممسوحة ضوئيا بـ

الغرض من الـ modularity هو اتاحة الفرصة لفصل أكواد الـ js لتنظيم الكود وتسهيل التعديل عليه .

عندنا two standards لتنفيذ الـ modularity

(١) الـ "commonjs module" وده الـ structure standard الخاص بالـ ES5 اللي بيوضحلنا ازاي ندمج structure الـ code js بتاعنا . باستخدام الـ require() .

(٢) الـ "module" وده الـ standard الخاص بالـ ES6

الـ commonjs ده الـ default standard .

ازاي نغير طريقة الـ modularity ؟

نحدد طريقة الـ modularity من خلال ملف الـ package.json باستخدام الـ property "type" ، القيمة الـ default للـ "type" هي "commonjs" .

الـ module ما هو إلا file js reusable ، هو ملف كل أقدر أعيد استخدام الأكواد اللي جواه في ملف كل تاني .

الـ module ES5 بنستخدم require

الـ module ES6 بنستخدم import from

https://www.linkedin.com/in/mohamadamir/

الممسوحة ضوئيا بـ CamScanner

١- ايه الفرق الجوهري ما بين الـ modularity من الـ ESS و ESG ؟
ايه الفرق بين الـ require و الـ import ؟

ESS module	ESG module
- مبيوزش default export	- بيوز default export مرة واحدة بس من الـ file الواحد
- مبيضيف جزء الـ "use strict" بالتالي مش هيظهر error لو استخدمنا variable من غير ما نعرفه	- بيضيف جزء الـ "use strict;" لوحده من غير ما اضيفه بنفسي.
- اقدر اعمل require لـ module جوا function أو if condition، اقدر احجم الـ scope بتاع الـ require	- الـ import لازم يبقى في الـ global scope مينفعش احطه في local block زي الـ if.

٢- ايه الفرق الجوهري ما بين الـ query() والـ execute() في الـ module mysql2 ؟
ليه الـ execute أحسن من الـ query ؟

- الـ execute أكثر أماناً من الـ query لانه بيستخدم كونسبت اسمه "prepared statement"، يعني الـ execute بيبعت الـ query للـ db server من غير data "أولاً" بعدين بيبعت الـ data في الـ request التاني بالتالي بيمنع الـ "sql injection".
لما الـ query بيبعت الـ query بال data في طول مرة واحدة.
- الـ execute أسرع لانه بيستخدم كونسبت اسمه (LRU) least recently used cache يعني لو عملت run تاني للـ statement هو مش هيعمل prepare من تاني.

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

الـ query بيبعت الـ query statement في خطوة واحدة للـ db server.

الـ execute بيبعت الـ query على خطوتين:
1) أول خطوة بيبعت الـ query للـ db server من غير data،
2) تاني خطوة بيبعت الـ data.

باختصار الـ execute بيعمل حاجة اسمها prepare يعني
جهزلي الـ statement دي عشان لما run اعملك الـ data.
بالتالي لو الـ data اللي اتبعتت فيها حاجة غريبة زي sql injection
أو الـ data فيها query جديدة فهو ساعتها مش هيعمل run.

في خطوة الـ prepare بتاعة الـ execute() احنا بنجهز
الـ query على قد الـ data اللي هتتبعت في الخطوة الثانية
بالتالي هنتضمن أن لسه في الـ data.

الـ execute بيستخدم concept اسمه LRU cache
الـ least recently used يعني بتكيش الـ statement
اللي نفذناها وسلسلنا preparation وتحطها في مكان ما، ولما نجي
نستخدم تاني الـ query تاني الـ execute هيجيبها من الـ cache
على طول من غير ما تعمل preparation من أول وجديد.

الـ execute أسرع من الـ query في الـ run الثانية، لأن
هو متضطر يعمل preparation لنفس الـ statement من أول
وجديد دلوقتي بتكيشها باستخدام الـ concept الـ LRU cache.

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

١٤- ايه هو الـ ORM ؟

هو تحويل الـ relational database لـ object oriented
بحيث نقدر نتعامل مع أي database باستخدام الـ oop بدلاً
من كتابة الـ code sql بشكل مباشر.

١٥- ايه ميزة الـ ORM ؟

١) أول ميزة انه بيشيل عني هم كتابة الـ query بنفسي، عن طريق
انه بيوفرلي أكواد جاهزة بتهندل كل query تبع الـ R-db.

٢) ثاني ميزة إني بكتب الـ query على شكل oop وليس sql
والـ ORM هو المسؤول عن تحويل الـ oop ده للـ sql المناسب
للـ dbms اللي بستخدمه.

٣) ثالث ميزة انه مبيعتمدش على RDBMS معينة، يعني
هو كود واحد بيشتغل على أي RDBMS.

١٦- اهم ميزات الـ ORM إني بكتب كود واحد بيشتغل على
أي RDBMS، بالتالي أقدر أغير الـ dbms اللي بشتغل عليه
من غير ما أغير في الكود بتاعي.

١٧- كل RDBMS ليه الـ version الخاص بيه من الـ sql
بالتالي الـ syntax ممكن يختلف من DBMS للتاني.

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

1- ال ORM اختصار لـ object Relational Mapping
object المقصود بيها أكواد oop .
relations " " " ال RDBMS .
mapping " " " انك بـ map أكواد ال oop بأكواد ال sql
الخاصة بال RDBMS اللي بتستخدمه .

2- ميزة الـ ORM ، 1) انك مش بتكتب ال query بايدك ، انما بتكلم function
جاهزة هي اللي بتكلم ال query .

2) بتكتب كود واحد بيشتغل على أي RDBMS .

3- ال sequalize دي اشهر ORM خاص بال nodejs .
ال mongoose دي اشهر ODM لل nodejs للتعامل
مع ال mongodb .
ازاي متستخدمهاش ! ع روح اقرا ال documents ☺

⑥) خطوات استخدام الـ sequalize :

١- اعمل install لـ "sequalize" ، واعمل install لـ db server اللي انت عايز الـ sequalize يعمل map عليه .

```
>> npm i sequalize
>> npm i mysql2
```

٢- اعمل connect بال db server ، والـ database اللي هتشتغل عليه ،

```
const {sequalize} = require('sequalize');
const sequalize = new Sequalize('اسم الـ db', 'الـ user', 'الـ pass',{
    host: 'localhost',
    dialect: 'mysql'
});
```

٣- • كده احنا عملنا الـ instance / الـ object اللي هيمثل connection بال db ، ولكن مش هيشتغل لوحده !! .
• عشان نـ start الـ connection لازم نعمل invoke الـ method .sync().
• لاحظ إن الـ sync() بترجع promise بالتالي لازم نتعامل معاها بـ async & await .

```
const connectDB = async ()=>{
    return await sequalize.sync()
    .then(result =>{console.log("DB connected", result);})
    .catch(err =>{console.log("fail to connect", err);});
}
export default connectDB;
```

٤- اول ما نعمل invoke لـ connectDB كده هتعمل start للـ connection مع الـ db بتاعتنا .

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

لازم نعمل export للـ instance اللي عملناه من الـ Sequelize
لأن الـ instance ده هو اللي بيحتوي الـ configuration بتاعة الـ db.
وهو اللي بيعمل start بين الـ connection مع الـ db دي.
وهو اللي بيحتوي على الـ methods اللي بنبعت بيها
queries للـ db اللي بيعمل connect عليها.

D: export sequelize;

٦- كل table هنستخدمه / هنعمله باستخدام الـ sequelize، هنعمله
ملف model خاص بيه.
← الـ table هو الـ model هو الـ schema.
← كل table من الـ db ليه ملف لوحده خاص بيه (tblname.model.js)
← كل table من الـ db بنعمله model أو/ schema باستخدام الـ sequelize.
← الـ table ده مش هيظهر في ملف الـ db إلا لما نعمل عليه أي operation
في البداية.

(ex) define 'Users' table:

① models/User.model.js
② import {DataTypes} from "sequelize";
→ import {sequelize} from "../DB/connection.js";
const UserSchema = sequelize.define('User', { }, { });
الـ options الـ attributes بتاعة الـ table.
export default UserSchema;

← ده object بيحتوي على الـ methods اللي هنبعت بيها queries / crud
للـ table "Users"، وبالتالي لازم نعمله export عشان نقدر
نستخدمه في الـ controller / router.

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

١- اول لما بتكريت ال table كده للموديل منتركي خلاص، مهما تعدل على ال schema مفيش حاجة هتتمسح من ملف ال db.
ليه؟ عشان انت مش ساح بال alter option لما عملت connect.

٨-
```
const connectDB = async ()=>{
    return await sequalize.sync({alter: true, force: true})
    .then( result =>{ })
    .catch( err =>{ } );
}
```

• ال property "alter" بيديني صلاحية التعديل على ال schema، يعني أي تعديل في ال table/schema/model هيتعمل في ال db على طول.

• ال property "force" بيمسح ال table اللي في ال db في كل مرة بتعيد تشغيل السيرفر، يعني كل لما السيرفر يقفل ويتعاد تشغيله من تاني.
← بيعمل drop لل db كل لما تعيد تشغيل السيرفر.

ملحوظة:

١ ال sync() دي اللي بتعمل fire للـ connection.
٢ كنترول ال alter / كنترول التعديل على ال table عشان يحقق ال db.
٣ دايماً اعمل false على ال alter option وال force option
• ولما تعدل على ال schema، شغل ال "alter"
• ولما تحب تمسح كل ال table في ال db وتعيد بناءها من الاول مع مضافة، شغل ال "force"

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

٩- الـ sequelize بيوفر methods معينة لتنفيذ الـ queries الخاصة بالـ relationship.

ولكن لازم تحدد علاقة كل table بالتاني، يعني تحدد علاقة الطرفين وليس طرف واحد.

```
(ex) UserSchema = sequelize.define('User', { }, { });
     ProductSchema = ~~~ ~ ~~ ~ ~~ ~~ ~ ~;
     UserSchema.hasMany(ProductSchema, {
         onDelete: "CASCADE",
         onUpdate: "CASCADE"
     });
     ProductSchema.belongTo(UserSchema);
```

لازم تحدد علاقة الطرفين بعض

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

week 5

1- noSql database Vs Sql database
2- mongoDB
3- data modeling / relation b/w collections
4- join b/w collections in mongoDB

https://www.linkedin.com/in/mohamadamir/

CamScanner الممسوحة ضوئيا بـ

noSQL vs SQL

١- من الاسبوع الرابع احنا اشتغلنا بالـ SQL database وعرفنا الآتي :

١) الـ SQL dbs بنكلمها من طريق الـ queries

٢) الـ query هي instruction معينة بتكلم بيها الـ db

٣) الـ SQL هي مش لغة برمجية ولكن هي لغة بنستخدمها بين الـ db والـ data اللي في الـ db .

٤) الـ database بينقسم لنوعين وهما :

- الـ relational dbs واللي هي الـ SQL db
- والـ non-relational dbs واللي هي الـ noSQL db

٢- الـ noSQL مش مقصود بيها no SQL ولكنها مقصود بيها not only SQL . no = not only

يعني هي بتاخد ميزات الـ SQL كلها وبتضيف عليها ميزات تانية .

٣- الـ noSQL مش احسن من الـ SQL والـ SQL مش احسن من الـ noSQL ،

احنا نقدر نعمل database بالاثنين ، لكن احنا بنختار الانسب للمشروع وليس الاحسن .

٤- الـ noSQL بتستخدم الـ distributed DBMS لأن الـ noSQL بيدعم الـ H-Scaling بالتالي نقدر نضيف عدد الاجهزة وننقسم الـ db عليهم .

٥- الـ noSQL يقدر يخزن كم مهول جداً من الـ data لانه الـ H-Scaling .

https://www.linkedin.com/in/mohamadamir/

الممسوحة ضوئيا بـ CamScanner

(1) Q] ايه الفرق بين ال SQL db Vs noSQL db ؟

SQL	noSQL
1) من حيث المسميات :	
ال table في ال SQL بيقابلها	collection في ال noSQL .
وال attributes/columns ~	fields ~ ~ .
وال records / ال data نفسها ~	documents ~ ~ .
* ال SQL بنسميها RDBMS	ال noSQL بنسميها distributed-DBMS وكمان دة مش معناه إن مفيش relations في ال noSQL وكمان ال relations بين ال data انسب في ال SQL db . واقوى
2) من حيث ال schema / ال structure بتاع ال record / ال data :	
• ال schema هي ال columns في ال SQL وال fields في ال noSQL .	
• ال SQL بيستخدم fixed schema يعني ال schema ثابتة مش بتتغير .	
• ال noSQL ~ ال schemaless أو ال dyamic schema .	
بالتالي ال noSQL انسب لتخزين ال unstructured data يعني ال data اللي ملهاش schema ثابتة / ملهاش structure ثابت .	
3) من حيث ال complex queries :	
• ال complex query هي query بتحتوي على sub-query عشان ال output هيطلعلك بعد ما يمر بكذا query مش مجرد query واحدة .	
• ال noSQL db صعب نعمل complex query ونطلع ناتج صح لأنها ودة لأنها مفيهاش relations واضحة بين ال data زي ال SQL .	

https://www.linkedin.com/in/mohamadamir/

الممسوحة ضوئيا بـ CamScanner

٤) من حيث أنواع الـ db : / أكثر من طريقة للتعامل مع الـ data

- الـ noSQL بيوفر أكثر من نوع / أكثر من طريقة لتخزين الـ data
- الـ noSQL بيوفر : (Radis) ① key-value database
  ② document-based database (mongoDB)
  ③ columnsbased database (Hbase)
  ④ Graph-based database (Graph-QL)

٥) من حيث الـ structure بتاع الـ db :
من حيث الـ scalability / إن احنا نعمل upgrade فيها .

- الـ SQL ماشية بـ vertical scale
ولكن تقدر تمشيها بـ horizontal scale ولكن
الـ implementation بتاع الـ SQL هيبقى ثقيل .

- الـ noSQL ماشية بـ horizonal scale
ولكن تقدر تمشيها بـ vertical scale زي الـ SQL ولكن
الـ implementation بتاع الـ noSQL هيبقى أثقل .

- الـ vertical scale أنسب مع الـ SQL db .
والـ horizontal scale أنسب مع الـ noSQL db
عشان الـ performance والـ implementation .

- الـ v-scale أسرع وأرخص وتنفيذه أسهل من الـ H-scale
ولكن الـ H-scale بيوفر unlimited upgrade .

المميزات والعيوب ؟

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

(2) @ اح الفرق بين ال scale Horizontal & vertical؟

ال scale المقصود بيه upgrade / زيادة امكانيات الجهاز اللي شايل ال db.

vertical scale	Horizontal scale
1) من حيث ازاي بينفذ / ازاي بيشتغل !	
• ال V-scale بيعني بنزود ال resources بتاعة الجهاز. يعني هو جهاز واحد بنعلي امكانياته.	
• ال H-scale بيعني بنزود عدد الاجهزة / ال servers اللي هتشتغل عليه، وعشان السبب ده بنسمي ال noSQL بـ Distributed-DBMS لانها بتدي support ال H-scaling.	
2) من حيث المميزات والعيوب !	
اتي استخدم أي واحد فيهم !	
• ال performance بتاع ال H-scale اسرع من ال H-scale وده بيدل ان ال SQL db اسرع من ال noSQL db.	
• ال implementation بتاع ال V-scale اسهل من ال H-scale لان كل اللي بعمله اني بزود امكانيات الجهاز ذات نفسه من جواه.	
إنما ال H-scale بنجيب جهاز جديد ونربطه بالقديم.	
• ال cost بتاع ال V-scale ارخص من ال H-scale.	
• ال upgrade limitation بتاع ال V-scale محدود عن ال H-scale يعني في ال V-scale بنقدر نعمل upgrade لغاية limit معين.	
إنما في ال H-scale بنقدر نعمل unlimited upgrade بالتالي ال noSQL بيقدر يتطور [illegible]	

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

Q(3) إزاي نقرر هنستخدم db SQL ولا noSQL؟

عشان نحدد الأنسب للمشروع، لازم نجاوب أسئلة معينة عن المشروع:

هي ستوبة أسئلة معتمدة أعتمد كلك مبادئ على المشروع بتاعي بيقدم إيه؟ وإيه الـ requirements بتاعته.

① هل الـ performance مهمة ولا لا، أهم الـ data تيجي صح وخلاص؟
← لو الإجابة yes يبقى هنستخدم db SQL
" " no " " db noSQL.

② هل هنعمل complex queries ولا لا؟
هل الـ data مترابطة بشكل قوي بالتالي هنقدر نعمل complex queries؟
← لو الإجابة yes هنستخدم db SQL.
" " no " " noSQL.

③ هل الـ schema ثابتة ولا احتمال نغيرها قدام؟
وهل التغيير في الـ schema هيعمل مشكلة ولا لأ؟
← لو الإجابة yes ثابتة ولو غيرنا مش هيحصل مشكلة يبقى هنستخدم SQL.
" " no يبقى هنستخدم الـ noSQL.

④ هل هنحتاج نعمل scale للـ db في المستقبل؟
هل هنتعامل مع كم مهول من الـ data في المستقبل؟
هل هنعمل scale بـ V-scale ولا H-scale؟
← لو V-scale هنستخدم db SQL
" H-scale " db noSQL.

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

١- ايه ميزات الـ noSQL؟

الـ noSQL بيدعم الـ schemaless لانه بيخزن الـ data في شكل object بالتالي هو انسب لتخزين الـ unstructured data.

- الـ noSQL يقدر يتعامل مع كم مهول جداً من الـ data لانه بيدعم الـ Horizontal scalling يعني أقدر أزود عدد الاجهزة اللي بشغل فيها الـ db.

٢- الـ noSQL بتسميته not only SQL لانه بيستعمل بديل من الـ db SQL العادية ولكن مش بيلغي ميزات الـ SQL.

٣- الـ noSQL بتسميته distributed DBMS لانه بيدعم الـ H-scalling بالتالي متقسم متقسم الـ db بتاعتنا على أكثر من جهاز / server.

٤- الـ noSQL مناسب لتخزين الـ unstructured data لانه schemaless.

٥- الـ v-scalling أسرع وأرخص وتنفيذه أسهل من الـ H-scalling ولكن الـ upgrade بتاعته limited

٦- الـ H-scalling بيوفرلنا ميزة الـ upgrade unlimited بالتالي نقدر نخزن كم مهول من الـ data.

https://www.linkedin.com/in/mohamadamir/

الممسوحة ضوئيا بـ CamScanner

٧- الـ db SQL عبارة عن جداول.
الـ db noSQL عبارة عن objects / documents

٨- الفرق بين الـ fixed schema و dynamic
ـ الـ fixed معناها إن انا أقدر أزود column أو field
على table عادي ولكن الفكرة إن الـ records القديمة لازم
أحطلهم قيمة بـ null للـ column الجديد اللي انضاف بالتالي
أنا حطيت data في الـ storage بتاع الـ db أنا مش عايزهم.
مثال كده اسم fixed يعني الشرط اللي هنورثه للنزل
بيجي على كل الـ table حتى لو عايزه على أجزاء منه فقط.

ـ الـ dynamic يعني أقدر أزود field على document أو الـ records
اللي أنا عايزهم فقط وليس الـ collection كل.

باختصار
- الـ fixed يعني التغيير على schema هيتم على كل
الـ records اللي عايزهم يتغيروا واللي مش عايزهم بالتالي
بخزن بيانات في الـ storage عالفاضي.
- الـ dynamic يعني التغيير على schema هيتم على الـ records
اللي عايزهم يتغيروا فقط.

- الـ fixed بيجي على مستوى الـ table كل.
- الـ dynamic ~ ~ ~ الـ record.

٩- الـ db relational بتتخزن في مكان واحد بالتالي أقدر أعمل
relations ما بينهم، أما الـ noSQL بتجي متقسمة على
أكثر ... اللي هو أعمل relation لكن مش متصل.

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

١٠- أقدر أعمل complex query لل SQL db والـ noSQL db
ولكن هتبقى أصعب وأبطأ في حالة الـ noSQL db لأن
مفيش relation واضح وقوي بين الـ collections زي الـ tables
في الـ SQL .

١١- الأهم اختلافات الأساسية بين الـ SQL والـ noSQL :
١) أن الـ SQL هو relational وموجود في جهاز واحد .
ولكن الـ noSQL db هو distributed ممكن يتفرقوا .

٢) الـ SQL هو Vertical والتاني horizontal .
٣) الـ SQL هو fixed schema والتاني dynamic .
٤) الـ SQL هو أنسب للـ complex queries .

١٢- امتى أستخدم الـ Relational database ؟
لو فيه ترابط بين البيانات هستخدم الـ R-db
، أما لو مفيش حاجة ترابطهم ببعض هستخدم الـ non-R

الخلاصة

① قواعد البيانات الـ SQL هي قواعد بيانات منظمة و
بتستخدم الجداول ، والبيانات فيها بتكون ثابتة
ومحددة الشكل (schema)

② قواعد البيانات الـ no-sql هي قواعد بيانات مرنة و
بتخزن البيانات بأشكال مختلفة : documents ،
graph ، key-value ، والبيانات فيها بتكون مرنة
وملهاش شكل محدد (schema-less) .

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

1) التعامل مع أي database هنحتاج حاجتين رئيسيين وهما:

① database server وهو الـ software اللي هيتحكم الـ db وهيـ execute الـ query statements ،

والـ db server ممكن تكون local أو cloud.

② database client وهو الـ software اللي هنـ connect بيه على الـ db server وبنبعت منه الـ queries.

---

db server زي الـ mongoDB

db client زي الـ mongoose

---

2) الـ mongoDB هو noSQL database بيخزن الـ data على شكل documents

---

3) الـ mongoose هو library زي بتسهلنا عملية connect على الـ mongoDB server وتعامل معاه من خلال الـ nodejs باستخدام لغة الـ js.

---

4) الـ mongoose بيشتغل بـ concept الـ ODM وده عبارة عن عملية تمثيل الـ tables / collections db على هيئة object بالتالي نقدر نتعامل مع الـ db كـ oop من غير ما نكتب أي queries بنفسنا.

---

5) الـ mongoDB URI هو الـ link اللي هنـ connect بيه على الـ mongoDB server.

(ex) mongodb://localhost:27017/b17

الـ protocol الخاص بالـ mongodb — الـ host اللي عليه الـ mongo — الـ port اللي بيـ run فيه — اسم الـ db اللي احنا عايزينه

https://www.linkedin.com/in/mohamadamir/

الممسوحة ضوئيا بـ CamScanner

٦- امتى نستخدم الـ mongoDB ؟

١- مع المشاريع اللي بتحتوي على كم كبير جدا من الـ data ،
٢- " " " بنتعامل مع unstructured data .
٣- " " " مالوش schema ثابتة وواضحة .
٤- " " " بتحتاج upgrade/scallability بشكل مستمر
٥- " " " بتحتاج real-time يعني بتحتاج سرعة عالية في تخزين وارجاع الـ data .

٧- امتى لا يفضل استخدام noSQL db ؟

مع الـ applications اللي بتحتاج تعمل transaction كتير ،
(الـ transaction عبارة عن مجموعة من الـ db operations اللي
هيتنفذوا مع بعض كـ units واحدة بالتالي لازم كلهم يتنفذوا
مع بعض أو يتلغوا كلهم مع بعض ، ميتنفذش حاجة وحاجة لا)

لو بنحتاج نعمل complex queries كتير في الـ application .

٨- كل table / collection عندك في الـ db متمثلة
بـ schema و model .
• الـ schema هو عبارة عن بنية الـ data اللي هتتخزن .
• الـ model هو الـ object اللي هتعمل بيه CRUD على
الـ collection اللي بيمثلها .
• يفضل كل model تتعمل في module لوحدها .

```
collection X = mongoose.Schema()
module.exports = mongoose.model('اسم', schema)
```

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

9- كل collection متسجل في الـ db منعمل schema و model
وهذه الـ object اللي بنعمل بيه CRUD على الـ collection
اللي بيمثلها.

10- إزاي في add نعمل create document ؟

1) save()
2) create()
3) insertMany()

(ca)

الـ save بيستقبل / بيضيف حاجة واحدة فقط في الـ request الواحد.
الـ insertMany بتضيف array of objects، أكتر من data.
الـ create بيضيف حاجة واحدة فقط في الـ request الواحد
الزي ؛
الـ create بتقدر اعمله call على طول من الـ model ، أما
الـ save بنعمله call من الـ instance بتاع الـ model.
يعني الـ create بيعمل instance من الـ model وبيعمل save
بشكل automatic.

الـ id هو field بينضاف بشكل تلقائي لكل document
بنعمله / بنضيفه والـ type بتاعه هو ObjectId.

الـ id نوعه ObjectId
بيتكون من حروف وأرقام
بيتكون من 24 hexadecimal يعني 24 حرف بال hexa.
بينضاف بشكل تلقائي لأي document.

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

١٣. ازاي ب get حاجة من ال db؟
ازاي ب search عن حاجة في ال db؟

1) find ()
2) findOne ()
3) findById ()

ال find بترجع كل حاجة في ال collection لو مش مبعت condition
والا هيعمل find لكل حاجة اللي بتـ match ال condition .

ال findOne بيعمل find حاجة واحدة بس لو مبعت condition
وإلا هيرجع أول حاجة بيلاقيها في ال collection .

لو عايزين ندور من خلال ال field id_ بقى نستخدم ال
findById () لأنه في الحالة دي هيرجع أسرع من ال findOne .

ال findOne بندور بيه على ال field ال unique اللي مش بيتكرر،
وإلا إيستخدم find .

١٢ ال create() وال save() احنا بنبعت data واحدة في ال request
الواحد.
ال insertMany احنا بنبعت أكتر من data في نفس ال request .

ال create بنادي من ال model على طول.
ال save لازم نعمل instance من ال model الأول.

١٤ كل ال crud methods في ال mongoose بترجع promise
ميعنى بتشتغل asynchronously بالتالي لازم نعمل async await .

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

(data modeling)

١٤- ازاي نمثل relation بين two collections؟
سواء كانت parent-child relation او child-parent؟

إذكر: إن collection هنعمله schema , model
في ملف module لوحده.

اولاً هنعمل الـ two collection.
ثانياً نحدد نوع الـ relation اللي جايزين نعمله.
احنا عندنا نوعين من الـ relations:
① child-parent ودي معناه إن الـ child عنده field
بتشيل كل الـ parents بتوعته، يعني الـ parent مش
عارف مين الـ childs بتوعته.
② parent-child ودي معناه إن الـ parent عنده field
بتشيل كل الـ childs بتوعته، يعني الـ parent عارف مين
الـ childs بتوعته.
ثالثاً: هنعمل field في الـ collection خاص بعملية الربط، والـ field
دي هيحتوي على two parameters اساسيين وهما:
① الـ type ودي لازم يكون ObjectId.
② الـ ref ودي هيحتوي على اسم الـ model / الـ collection
اللي هنعمل relate بيه.

```
Product.
{ _id:,
  title:,
  createdBy:{
      type: ObjectId,
      ref: 'User'
  }
}
```

```
User
{ _id,
  name,
  email,
}
```

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

١٥- في الـ db الـ noSQL احنا بنستخدم مصطلح الـ modeling وليس مصطلح الـ relations لأن احنا مش RDBMS.

الـ data modeling هي عملية تشكيل الـ structure بتاع الـ data.

١٦- الـ collection بتتكون من schema و model.

١٧- في الـ noSQL مبنعملش relation ولكن بنعمل modeling

١٨- عشان نعمل relation / data modeling ما بين two collection ، بنعمل reference في collection بيشاور على الـ collection التاني ، (هو شبه فكرة الـ FK)

١٩- الـ reference لازم نحدد له نوعه وهيشاور على مين :
① نوعه لازم يكون نفس نوع الـ PK / الـ id.
② هيشاور على الـ collection اللي عايزين نربط بيه.

٢٠- عشان نربط two collection ببعض ، هنعمل field في collection منهم يكون الـ reference للـ collection التاني.
مواصفات الـ reference field هتكون كالآتي :
① اسمه يفضل يكون معبر عن نوع العلاقة.
② نوعه لازم يكون نفس نوع الـ id ، لأنه هيخزن قيم الـ id.
③ لازم نحدد فيه reference على انهي model / collection.

٢١- عشان نجيب بيانات من two collection ، هنستخدم الـ populate() method وده هنحط فيه اسم الـ reference field اللي هنعمله بس.

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

٢٣ ال populate() بتسهلنا نجيب document من collection تانى، هو بيعمل replace لل path اللى فى ال document بال document اللى بيه reference ملزق من collection تانى.

```
.populate({
    path: ' ,  اسم ال field اللى بيه reference فيه
    select: ' اسم ال fields اللى عايزها تتعرض
});
```

٢٤ ال populate() بتستخدم مع ال find().
ال populate بتستخدم عشان نجيب/نعمل get لل document من collection تانى من خلال ال field اللى بيه reference فيه.

٢٤ ازاى نجيب document من collection تانى؟
ازاى نعمل join من two collections؟
ازاى نجيب data من two collections؟

عندنا طريقة ال populate()
وبرضه ال aggregate() باستخدام ال $lookup:{}

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

week 6

1- http status code
2- environment variables
3- hashing - encryption
4- Error handling
5- Json web Token

https://www.linkedin.com/in/mohamadamir/

CamScanner الممسوحة ضوئيا بـ

http status code

1- الـ status code ما هو إلا رقم بيعبر عن حالة الـ request.

2- الـ status code ما هو إلا رقم بيتبعت مع الـ response بيدل أو بيعبر عن حالة الـ request اللي اتبعت.

3- أي status code بيتكون من 3-digits وأول digit بيعبر عن الـ category اللي بيتبعها الـ code.
الـ status code متقسم لـ 5-categories من 1 لـ 5.

4- الـ status code لو بدأ برقم 1 يعني بيعبر عن حاجة في الـ network نفسه.
~ ~ ~ 2 ~ ~ ~ نجاح العملية.
~ ~ ~ 3 ~ ~ ~ deep redirection.
~ ~ ~ 4 ~ ~ ~ error من ناحية الـ client.
~ ~ ~ 5 ~ ~ ~ error ~ الـ server.

5- الـ status code هو رقم بيتبعت من الـ backend للـ frontend
عشان الـ frontend يفهم إيه اللي حصل للـ request اللي بعتته.

6- 3xx بيعبر عن الـ redirection يعني الـ request اتحولت إعادة توجيه من الـ server لـ server تاني لسبب ما، سواء علشان عطلات أو رفعنا الـ app في سيرفر تاني ...

7- في 3rd party module اسمه "http-status-code" وكل فكرته إنه بدل ما نكتب رقم الـ status code بإيدينا، احنا هنكتب كلمة أو variable يحتوي على الرقم ده.
يعني كل ميزته إنه بدل ما نكتب رقم، هنكتب اسم يحتوي الرقم ده.

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

env, dotenv

١- الـ env file هو ملف بنشيل فيه القيم الـ private اللي
مينفعش أي حد يشوفها في الكود زي الـ key number اللي بالتاسع
أو الـ URI mongo اللي مكتوب فيه الـ username والـ pass ..

٢- الـ environment variables ما هم إلا private variables
بنسميها private values / secure values / sensetive values .

٣- إيه الغرض من ملف الـ env ؟

بنشيل فيه الـ sensetive values اللي مينفعش أي حد يشوفها
في الكود .
ملف الـ env هو ملف بنشيل فيه private variables وبنستخدم
القيم اللي فيه من خلال اسم الـ variable وليس القيمة بشكل مباشر .

٤- إزاي نعمل env file وإزاي نتعامل معاه في الـ node ؟

هنستخدم 3rd party module اسمه "dotenv"
١) اعمل ملف ".env"
٢) نزل الـ dotenv module ← npm i dotenv
٣) اعمل import للـ dotenv module في الـ index file عشان يبقى متشاف
على مستوى الـ app كله ..

```
import * from 'dotenv';
dotenv.config();
```

٥- من خلال الـ env file نقدر نستخدم private values
بشكل غير مباشر في الكود من خلال اسم الـ variable
اللي بنشيل .

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

٦- ايه هو ملف ال env. وايه لازمته؟

هو عبارة عن private file بنشيل فيه ال private values ، القيم الحساسة اللي مينفعش نكتبها بشكل مباشر في الكود
ازاي؟ من خلال إن احنا نشيلهم في ال env file على شكل Key = Value ولما نحتاج نستخدم ال value ، هنكتب ال key اللي بتاعها وهيرجع القيمة بشكل مباشر.

⇐ هو عبارة عن private file بنشيل فيه ال private values اللي البرنامج مينفعش يشتغل من غيرها ومينفعش نكتبها في الكود بشكل مباشر.

هو ملف مخفي في السيرفر بنشيل فيه ال sensitive data اللي مينفعش البرنامج يشتغل من غيرها ومينفعش اي حد يشوفها في الكود ، مينفعش القيم دي تظهر في الكود بشكل مباشر.

٧- ال dotenv.config() بتبحث ال path ، لكان ملف ال env. هتحاول تجيب القيم اللي فيه جوا ال process.env بالتالي لما تستخدمهم في سيرفرك ال node والا مش هتعرف تستخدمهم لأن ملف ال env هو مش ملف ده أصلاً.

لو ملف ال env موجود جنب ملف ال index أو جنب الملف اللي بتعمل فيه ال dotenv.config() ، إذا مش لازم ابحث ال path بتاع ال env file في ال config() ، هو هيحطه بنفسه.

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

الخلاصة يا بيدو كالآتي:

ملف الـ env هو ملف مخفي من السيرفر (مخفي يعني محدش يقدر يوصله إلا من السيرفر نفسه) الملف ده كل وظيفته انه يشيل القيم الـ sensitive اللي البرنامج ميقدرش يشتغل من غيرها وميفضلش حد يشوفها في الكود بشكل مباشر.

ملف الـ env بيخزن القيم على شكل key=value وإحنا بنستخدم الـ value بشكل غير مباشر في الكود من خلال الـ key بتاعها.

ملف الـ env. دي مش ملف js بالتالي عشان نقدر نستخدمه في الـ nodejs بنستخدم 3rd party module بيقرأ الـ env variables اللي في ملف الـ env جوا الـ process في object اسمه env.

الـ dotenv دي 3rd party module بيخلينا نقدر نعمل access لملف الـ env من الـ process بتاع الـ nodejs اللي شغالة.

https://www.linkedin.com/in/mohamadamir/

الممسوحة ضوئيا بـ CamScanner

(hashing & encryption)

1- ايه الفرق بين ال plaintext وال ciphertext؟

ال ciphertext هو النسخة المشفرة من ال plaintext ، الغرض من التشفير ان احنا نخلي ال plaintext يبقى unreadable للناس اللي مش عاوزين مفتاح التشفير.

الغرض من التشفير هو حماية بيانات ال users.

2- ايه الفرق بين ال hashing وال encryption؟

الاثنين عبارة عن algorithm للتشفير، الفرق بينهم إن:

- ال hashing بيشتغل one way يعني بيحول الداتا ومقدرش يرجعها لأصلها.
- ال encryption two way يعني بيقدر يحول الداتا ويرجعها لأصلها تاني.

- ال hashing مقدرش يرجع الداتا لأصلها ولكن بيقدر يعمل compare.
- ميزة ال hashing انه آمن من ال encryption
- ميزة ال encryption ان أقدر انزل التشفير من خلال ال private key وال key ال public اللي بشاركه مع اللي انا عايزه.

3- هل ال password نعمله hashing أو encryption؟

نعمله hashing طبعاً لأنه هيبقى آمن للـ users لأن الداتا اللي اتعملها hashing مقدرش نرجعها تاني.

4- امتى نستخدم hashing وامتى encryption؟

لو الداتا اللي انا عايز اشفرها لو محتاج نرجعها لأصلها تاني يبقى نستخدم encryption

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

الـ password هنشفرها بـ hashing
الرقم الشخصي ~ بـ encryption
رقم البطاقة الشخصية ~ بـ encryption وهكذا ...

٥- ليه بنحتاج نشفر بيانات الـ user ؟

عشان نحمي بيانات العملاء ،
عشان نضمن سرية بيانات العملاء بتوعنا ،
عشان لو البيانات دي اتسربت تمكن العميل يقاضينا فيل .

٦- مين اللي بيعمل الـ algorithms بتاعة التشفير ؟

دي شغلانة الناس بتوع الـ cyber security ، احنا مجرد بنستخدم الـ developer .

٧- ازاي نعمل hashing في الـ node ؟

هنستخدم 3rd party module اسمه 'bcrypt'
هنستخدم method اسمه hashSync() بيستقبل مني حاجتين :
① الـ plaintext اللي عايزين نهشفره .
② الـ saltround ودي عدد المرات اللي عايزين نعمل hashing ورا بعض .

• كل ما نزود الـ saltround ، كل ما زاد قوة الـ hashing ، كل ما زاد وقت التشفير ، كل ما قل الـ performance بتاع الـ app .

• كل saltround بتنفذ ضمنياً ٤ مرات يعني لو كتبنا الـ saltround بـ 5 مثلاً ، كده الـ hashing هيتنفذ 5 * 4 = 20 مرة .

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

٤. ايه الفرق بين bcrypt module و bcryptjs module

الـ bcrypt محول لـ nodejs بالتالي هتحتاج nodejs عشان تشغله،
الـ bcryptjs محول بالـ js plain العادية بالتالي هتقدر تستخدمه مع
أي لغة js من غير ما تحتاج nodejs، عكس بتشتغل على الـ browser على.

كل ما زادت الـ saltround، كل ما زاد قوة الـ hashing
، كل ما زاد الـ security، كل ما زادت وقت عملية التشفير
، كل ما قل الـ performance بتاع الـ app.

اتفق رقم الـ saltround هو ما بين 8 و 10 هو الوسط.
8 عشان لو زاد هيقلل الـ performance، ولو قل هيقلل من الـ security.

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

Error handling

١- اي error بيحصل من الـ backend هيوقع السيرفر
مع عكس الـ frontend بالتالي من الـ backend لازم نهندل
أي error ممكن يحصل .
ليه لازم نهندل الـ errors؟

٢- احنا لازم نهندل الـ error عشان ميوقفش الـ runtime ،

٢- ازاي نهندل الـ errors من الـ كود ؟

عندنا طريقتين :

(١) طريقة الـ try-catch من طريقة الكود اللي شايفين انه ممكن يعمل
error مخطئ من الـ try block بحيث الـ try هيجربه لو اشتغل
تمام ولو عمل error الـ catch block هيمسكه وباقي الكود هيكمل
من غير ما يقف .

(٢) طريقة الـ promise rejection handling وديه بنستخدمها مع الـ promise
فقط باستخدام الـ catch method .

٤- ايه مشكلة / عيوب الـ try-catch blocks ؟ ايه الحل ؟

مشكلة الـ try - catch ان جزء الـ catch بيبقى doubled
متكرر وده هيصعب علينا التعديل عليه وهيقلل الـ readability
بالتالي احنا عايزين طريقة نوصل بيها كل الـ response بتاع
الـ catching error .
الحل ان احنا نعمل function نبعتلها الكود اللي عايزين نهندل أي
error فيه وهي تهندله ، بحيث هيكون هو كود واحد هو نفسه الكود
هنهندل بيه أي error ، بالتالي هنحل مشكلة الـ readability والـ
maintainability بتاعة الـ try - catch .
الحل نستخدم الـ middleware asyncHandler .

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

٥- مشكلة ال try-catch ان الجزء بتاع ال try دمو اللي بيتغير
لما جزء ال catch بيبقى ثابت من كله وكده احنا بنعمل
double code ، تكرار لنفس الكود وده بيصعب عملية التعديل
لان هنضطر اعدل نفس الكود من كل الاماكن اللي استخدمته فيها .

الحل ان احنا نعمل function نبعتلها الكود اللي هيحط في ال try
كـ parameter لان ده الجزء ال dynamic اللي بيتغير، وجزء ال
catch هيفضل ثابت من كله .

(ex)

```
function catchError(fn){
    try{ //dynamic part
      fn()
    }catch(error){
      //static part
    }
}
```

المشكلة هنا ان ال fn
مش هتقدر اتبعتلها argument
، ازاي الحل ؟

الحل ان احنا نخلي ال catchError يعمل return لـ function
وده بسبب ميزة ان ال js من higher order function
بتعامل ال functions كـ first class objects .

```
function catchError(fn){
    return (x)=>{
        try{ fn(x) }
        catch{ }
    }
}
```

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

٦- إيه الفرق بين وظائف ال middleware وال helpers؟

ال middleware بتخطط في ال controllers أو الأكواد اللي بتشتغل في ال sequence أو ال pipeline بتاع ال router

ال helpers هي مجموعة functions بتستخدم في أي مكان في الكود.

٧- في ال express الأكواد اللي بتشتغل جوا ال try-catch هي أكواد ال controller / ال routerHandler، ولكن زي ما عرفنا فيه طريقة ال try-catch block هو تكرار جوا ال catch. بالتالي هنشغل function نوصل بيها ال response بتاع ال catch handling.

```
export const catchError = (controller)=>{
    return (req,res,next)=>{
        controller(req,res,next).catch(error=>{})
    }
}
```

٨- إيه هو ال asyncHandler؟

هو middleware بتعمل catch لل error اللي بيحصل في ال controller بتاع ال express router وبتبعت ال error ده لل global error handler اللي في ال express أو لل custom error handler.

ميزته إنه بيحل مشكل ال try-catch block، هو بديل لل try-catch ميزته إنه بيشتغل مع ال promise code فقط.

https://www.linkedin.com/in/mohamadamir/

الممسوحة ضوئيا بـ CamScanner

9- إيه هو الـ express error handler ؟

ما هو إلا middleware عادي جداً الفرق إنه بيستقبل أربعة parameters وأول parameter فيهم هو الـ error .

دايماً الـ error handler بيكون آخر middleware مكتوب في الـ middleware stack .

لو عملناش custom error handler بنفسنا، الـ express هيستخدم built-in error handler تبعته وده الـ default اللي موجود في أي express app في حالة إن احنا عملناش واحد بنفسنا .

الـ error handler middleware بيشتغل لما بنبعت data في الـ next() method .

10- الـ next() method لو ضربت منك أي data ماما كلمة 'route' ، هتعمل skip أي middleware جاي وراها وتروح تشغل الـ global error handler بتاع الـ express أو الـ custom error handler لو اتعمل واحد، والـ data اللي اتبعتت هتكون قيمة الـ error parameter .

الـ next() لو ماضربش منك data، هيشغل الـ middleware اللي جاي بعده مباشرة، أما لو ضربت منك data هيعمل skip أي non-error middleware جاي بعده، وهيشغل الـ error middleware وقيمة الـ error parameter هي الـ data اللي في الـ next() .

https://www.linkedin.com/in/mohamadamir/

الممسوحة ضوئيا بـ CamScanner

11- ازاى تتعامل error handling فى الـ {express}

هنحتاج ٣ حاجات :

① نعمل asyncHandler : catch بتاع الـ error بدلا من استخدام الـ try-catch .

② نعمل custom error handler ينزل بتاع أى error اتعملت catch فى الـ application ،

③ هنبعت الـ error اللى اتعملت catch للـ error handler باستخدام الـ next() method .

```
(ex) const asyncHandler = (fn) => {
①      return (req, res, next) => {                    ③
          Promise.resolve(fn(req, res, next)).catch(err => next(err))
       }
}
```

- الـ promise.resolve() بتحول الـ value اللى جواها لـ promise ،
ولو الـ value أصلاً promise بترجع زى ما هيا ،
ولو الـ value هى thenable بتنفذ .then() ، الـ promise.resolve
هترجع الـ .then() دية .
- السبب الوحيد لاستخدام الـ promise.resolve هو تحويل الـ value
اللى ممكن تكون promise لـ promise (حول value لـ promise)
- الـ promise.resolve(value) طريقة مختصرة لكتابة
new Promise(resolve => resolve(value))

```
② const errorHandler = (err, req, res, next) => {

}
```

---

③ next(5) = errorHandler(5, req, res, next)

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

١٢. ازاى نشغل error object ؟ وايه محتوياته ؟

```
• let error = new Error('error message');
  console.log(error); // { name: ,
                          message('error message',
                          stack:
                        }
• next('invalid pass');
• next(new Error('Invalid pass'));
```

ال new Error() بيعملنا error obj بيحتوى على name و message, stack.

ال stack parameter بيحتوى على معلومات عن المكان اللى حصل فيه ال error وده ميفضلش نبعته لل frontend / لل user and إحنا بنحتاجه اثناء ال development فقط.

• لو عايزين نعرف بيانات أكتر عن ال error كـ developer هنستخدم ال err.stack parameter.

• لو عايزين نبعت بيانات أكتر عن ال error لل user هنستخدم ال status code.

ال status code هو رقم بنبعته لل frontend، والرقم ده بيعبر عن حالة ال request اللى اتبعت، هل نجح ولا فشل وفشل ليه

١٢ ازاى نشغل ال error middleware؟ وازاى نبعت ليه قيمة ال error؟

كل اللى هنعمله من ال invoke ال next method ونبعت قيمة ال error object

next(new Error(...))

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

1. ايه هو ال web auth؟

ال web auth مقصود بيه ال authentication web. وال authorization web.

2. ايه الفرق بين عملية ال authentication وال authorization؟

ال authentication هي عملية تسجيل ال user في الموقع عن طريق ال registeration وال login، والهدف منه إن أعرف مين ال user اللي بعت ال request.

ال authorization هو إن أحدد لل user ايه اللي يقدر يعمله في ال system، ايه صلاحياته.

ال authorization بيتعمل بعد ال authentication، يعني قبل ما حدد صلاحيات ال user لازم أعرف مين هو ال user أصلاً.

3. ايه هو ال tolkin؟

ال token عبارة عن معلومات معينة عن ال user اللي عمل login. ولكن المعلومات دي بتكون متشفرة، والمعلومات دي بتكون insensetive يعني مفيش معلومات تضر بأمن ال user.

الغرض من ال token إن أعرف ال user/ال system إن ال user اللي باعت ال request عامل login بالفعل فيضطر يعمل login في كل مرة يبعت فيها request.

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

٤- ايه مكونات ال token ؟

ال token هو بيانات ال user اللي عامل login
ولكن متشفرة ، وأي token بيتكون من ٣ اجزاء .

① جزء ال header وده بيحتوي على نوع ال token ونوع
ال algorithm اللي اتعمل بيه التشفير .

② جزء ال payload وده ال data اللي شفرتها .

③ جزء ال signature وده ناتج تشفير ال header وال payload
بال base64 ، ومضاف اليه secret key وده رقم سري
موجود في السيرفر فقط محدش يعرفه إلا ال backend
وبيستخدم كلمة السر دي عشان ف generate ال token
وبيستخدمه عشان يعمل verify وال decode ال token .

- ال signature هي كلمة سر ميعرفها شي إلا ال backend
- ال signature دي اللي بتعمل generate بيه ال token وبيعمله verify بيه .
- ال signature بيتكون من ال header وال payload متشفرين بال base64
ومتنور عليهم ال secret key ، بالتالي أي تغيير في ال token
هيعمل تغيير في ال signature بالتالي مش هيعمله verify .
- كل user عامل login ، بيعمله token خاص بيه .
- ال token بيفضل valid إلى ما الانتهاء لغاية ما نحدد له exp date
- لو غيرنا/عبثنا في ال token ، ال signature اللي فيه هيتغير .
- ال token هو المفتاح اللي بيأكد إن ال user عمل login .
- ال token فيه منه نوعين : ① normal token
② Barer token

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

٦- ليه ظهر concept الـ token؟
أي المشكلة اللي أدت لظهور فكرة الـ token؟

المشكلة هي إن الـ http عبارة عن stateless protocol
يعني كل request بيتعامل كـ request جديد أول
مرة بيتوقف ومش هيفتكره.

بالتالي كل request بنبعته لازم نبعت معاه الـ login credentials
زي الـ pass والإيميل عشان نعرف الـ server إن احنا user معين

المشكلة باختصار إن احنا بنظر نعمل authentication مع كل
request بنبعته عشان نعرف الـ server احنا مين لأن
الـ http هو stateless protocol.

الحل إن احنا نبعت بيانات الـ user اللي عمل login بشكل
مشفر للـ frontend، ونبعت الشفرة دي مع كل request
بحيث هتكون المفتاح اللي هيأكد للسيرفر إن الـ user عامل login.

٧- إيه الـ process بتاعة الـ token-based auth؟

١) الـ user هيعمل login من الـ system.
٢) الـ system هيتأكد من الـ login ويعمل token خاص بالـ user
ويبعته للـ client-side / frontend.
٣) الـ frontend هيخزن الـ token عنده في الـ cookie أو الـ localstorage
٤) هنبعت الـ token مع كل request عشان نفكر الـ server
إن إحنا عاملين login.

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

- ليه الأماكن اللي بنبعت فيها الـ token من الـ request؟

احنا نقدر نبعت الـ token في أي مكان في الـ request
سواء في الـ body أو الـ query أو الـ url، إنما بنحط
نبعته في الـ request Header ليه؟
لأن الـ Header هو المكان المشترك بين كل الـ http methods
بالتالي هنوحد المكان اللي هنبعت فيه الـ token تماما.

- ليه هو الـ jwt؟
هي library بتعمل encode و decode لـ data
معينة، وبتستخدم secret key عشان تعمل التشفير.

الـ jwt secret key هو كلمة السر اللي بتعمل encode أو
generate للـ token وبتعمل decode للـ token.

- الـ token هي جزء بتأكد للسيرفر إن الـ user اللي عامل
الـ request عامل login قبل كده.
بنحتاج الـ token لأن الـ protocol هو http protocol
stateless يعني بيتعامل كل request على إنه request جديد أول مرة
بيعمله.

- الـ signature هو أهم جزء في الـ token لأنه من خلاله
هنقدر نعمل verify للـ token ونعمله decode، لأن
الـ signature بيحتوي على الـ secret key اللي عملنا بيه generate
للـ token، واللي من خلاله هنقدر نعمل decode للـ token.

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

١٢ ايه هو الـ Bearer token ؟

ما هو إلا كلمة بنخطها / بنزودها قبل الـ token
الغرض منها انها تزود لي الـ security بتاع الـ token
او انها تحدد نوع الـ signature اللي هعمل بيه verify
لو انا موفر أكثر من token مختلف .

ليه الـ Bearer token ظهر ؟

ذكره الـ Bearer token ظهرت لما ابتدينا نستخدم الـ Oauth
وهو تسهيل الدخول باستخدام الـ social account زي google، facebook...
بالتالي بقيت اعمل generate لـ token للـ auth العادي بتاع الموقع .
و ~ ~ ~ ~ للـ auth بتاع الـ facebook
و ~ ~ ~ ~ للـ ~ ~ الـ google ... وهكذا

امتى نستخدم الـ Bearer Token ؟

(١) لو عندنا أكثر من نوع token وكل نوع بنعمله verify
بطريقة مختلفة أو بـ secret key مختلف ، بالتالي هنحط كلمة معينة
قبل الـ token اللي هنبعته في الـ request ومن خلال الكلمة دي هنعرف
انهي نوع token هنشتغل عليه .

(٢) لو عايزين نزود الـ security بتاع الـ token بحيث نحط كلمة
سيكريت مش غير الـ frontend والـ backend ولو الكلمة دي متبعتش
قبل الـ token ده معناه إن الـ token ده أخذه وبعته من موقع تالت
أو من جهاز تالت (ولكن الطريقة دي مش مفيدة أوي لأن احنا
عندنا الـ cors origin)

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

week (7)

1- sending email (nodemailer).
2- confirm email
3- validation (joi)
4- file upload (multer)
5- Qrcode
6- cron job

https://www.linkedin.com/in/mohamadamir/

CamScanner الممسوحة ضوئيا بـ

(sending email)

1- sending email يعتبر من التاسكات الأساسية عند الـ backend dev اللي لازم تعرف تعمل.

2- عشان نبعت email لازم يكون فيه مكان هو اللي هيبعت الإيميلات دي، والمكان دي بنسميه service، الأماكن دي زي (gmail، yahoo، outlook)

3- ليه بنحتاج نبعت email؟

1) علشان نعمل verify لل email بحيث نتأكد ان ال email ده موجود فعلاً أو سليم.
2) علشان نبعت target password لل user
3) علشان نعرف نبعت أي معلومة لل user زي عرض أو update أو خبر...

4- ازاي نبعت email بال nodejs؟

هنستخدم 3rd party module اسمه "nodemailer".
نعمل npm i nodemailer

5- ازاي نستخدم ال nodemailer؟

هما خطوتين لا أكثر ولا أقل:
① أول خطوة بنعمل create لحاجة اسمها transporter وده object بيحتوي على ال configuration الخاصة بال email أو ال email service اللي هنبعت بيها الإيميلات.

② ثاني خطوة بعد ما جهزنا الإيميل/service اللي هنبعت بيها الإيميلات

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

٦- الـ transporter بيستقبل مني،
الـ host والـ port الـ الايميل بتاعنا موجود فيه.
أو الـ service الـ التحميل بتاعنا موجود فيه لو الـ server الـ
رافعين عليه الـ project مش موفرلنا email خاص بينا.

الـ auth ودي بنحط فيه الـ ~~criteria~~ البيانات الخاصة بتسجيل
دخول الخاصة بالايميل زي اسم الايميل والـ password، ودي منطقي
كانه مينفعش نستخدم الـ service gmail مثلاً إلا لما نسجل دخول
فيه ودي منطقي ☺.

٧- الـ transporter ممكن يكون ايميل السيرفر الـ انا رافع فيه
الـ project لو كان متوفر، وممكن يكون mail service زي gmail
أو outlook.

٨- ايه مشكلة الـ auth؟

اولاً عشان نقدر نستخدم الـ mail service ايا كانت هيا ايه؟
لازم اولا نسجل دخول فيها ودي شيء منطقي.

المشكلة إن الـ password لوحده قدر يوصلك كتير جدي
فيه حطوا عليه بيانات تانية، اكل ان شركة google لغت
استخدام الـ personal password بتاع الـ gmail واستبدلته
بالـ app password، ودي password بتدي للـ app الـ هيستخدم
الـ email ويكون مختلف تماماً من الـ personal password.

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

مراجعة سريعة (ازاي تبعت email بال node.js)

١) هنستخدم nodemailer module.

٢) هنجهز الـ transporter ، يعني هنجهز الـ email اللي الـ application هيبعت بيه الايميلات.

٣) هنبدأ نبعت الايميلات مع ملاحظة ان محتوى الايميل اللي هنبعته هيكون كالآتي :

from : الايميل الخاص بالـ app

to : ايميل الـ user اللي عايزين نبعتله

subject : عنوان الرسالة

text : الرسالة كنص

html : الرسالة كـ html

الرسالة اللي هنبعتها كـ html بنسميها الـ email template.

الـ email templates من أكتر الـ tasks اللي هتستخدمها في لية ؟!

لأن الـ email client سواء كان desktop أو mobile أو web app كل واحد منهم بيكون ليه الـ rendering engine الخاص بيه بالتالي مش كل أكواد الـ html والـ css هتكون مدعومة.

الـ email client هو الـ software المسؤول عن ارسال واستقبال الايميلات وده ممكن يكون desktop أو web أو mobile كل واحد منهم بيكون ليه rendering engine خاص بيه بالتالي مش كل أكواد الـ html والـ css هتشتغل عند كله.

https://www.linkedin.com/in/mohamadamir/

الممسوحة ضوئيا بـ CamScanner

Task for you

Task 1 : دور على youtube ازاى تبنى قالب email
How to build html email template?
The right way to build HTML email template?

Task 2 : دور عن ليه زعمى ال html email template
بيه render بشكل مختلف فى كل ال email clients ؟

Task 3 : ايه هو ال email server وال email client ؟
وليه كل email client بيه render ال html بشكل
مختلف عن التانى ؟

---

② ليه زعمى ال html email template بيه render
بشكل مختلف لكل email client ؟

لسببين :

① اول سبب خاص بال security وهو ان ال mail service
بيعمل pre-processing لل html بحيث بيحذف الاكواد اللى
ممكن تعمل مشكلة ك security زى ال js .

② ثانى سبب هو ان كل email client بيكون ليه render engine
خاص بيه بالتالى مش كل اكواد ال html هتشتغل فى كل
ال email clients .

Task 4 : how to implement confirm email functionality?
[illegible] nodemailer?

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

validation

١- الـ validation هو أن أحمي الـ data الـ جاية

٢- الـ validation الغرض منه هو التأكد من صحة وسلامة الـ data الـ جاية من الـ client.

٣- ليه validation؟ لسببين:
① أحمي الـ site بتاعك من أي attack / hack.
② أتأكد إن الـ user دخل data صحيحة الـ أنا عايزها.

٤- الـ validation أحد أهم النقط الأساسية في أي application سواء front أو back والأهم طبعاً هو جزء الـ backend لأنت الجزء الأساسي الـ بيتعامل مع الـ data وبنقدر نبعتلة الـ data حتى بدون الـ frontend من خلال postman مثلاً.

٥- ألتزم ولابد أعمل validation على كل input في الـ website

٦- ألتزم أتأكد من الـ data الـ جايالنا قبل ما نستخدمها في الكود عشان نتأكد من حاجتين وهما: الـ data آمنة مفيهاش hacking code. والـ data صحيحة مفيهاش أخطاء.

٧- إزاي نطبق الـ validation؟ إن إحنا نحط شوية rules على الـ input ونتأكد إذا كان حققت الـ rules دي ولا لأ.

٨- الـ zod هي أشهر validation library للـ zod، مزيتها إن إحنا نقدر نستخدمها مع الـ frontend أو الـ backend.

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

٩. ازاي الـ joi بيشتغل؟

← اولاً لازم نبني الـ schema بالشكل اللي انا متوقع الـ data تيجي بيه، بعد كده بنقول validate بالـ schema على الـ data اللي جت فعلياً من الـ client وبناء نشوف لو فيه error ولا لا.

← هنا ٣ خطوات، (١) اعمل الـ joi schema
(٢) استقبل الـ data من الـ client
(٣) validate عمل.

١٠- ازاي الانا كمان اللي عكس بنجيب نبعت الـ data للـ server؟
~ ~ ~ ~ ~ ~ ~ ~ للـ backend؟
الـ req header، الـ req body، الـ param، الـ query.

(ex)

① ~~const signUpValidationSchema = joi.object({~~
~~body: { name: joi.string(),~~
~~}~~
~~params: { name: Ajo~~

① const schema1 = {
body: joi.object({ })
params: joi.object({ })
headers: ...
}

② const data = req.body;

③ const result = joi.validate(data, schema1.body);
(req.body, schema1.body);

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

validation.middleware.js

```
const reqKeys = ['body', 'headers', 'query', 'params'];
```

دية الاماكن اللي الـ data ممكن تكون موجودة فيها

```
export const validationMiddleware = (schema) => {
    return (req, res, next) => {
        let validationErrorArr = [];
        for (const key of reqKeys) {
            const validationResult =
                schema[key]?.validate(req[key], {abortEarly: false})
            if(validationResult?.error){
                validationErrorArr.push(...validationResult.error);
            }}
        if(validationErrorArr.length){
            return res.status(400).json({
                err_msg: "validation error",
                errors: validationErrorArr
            });
        }
        return next();
    }
}
```

متخرجش على أول error
ملاقيه ولكن كل باقي
الـ errors

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

ازاى نعمل rule joi custom

- الـ rule هى الـ validation أو الحاجة اللى بنعملها validate.
- الـ validation هو rule أو مجموعة من الـ rules بنطبقها على field معين
- input
- الـ joi بيوفرلنا rules جاهزة ،
- عشان نعمل rule custom هنعمل الآتى ،

① أولا هنجهز الـ rule وهى عبارة عن function بتستقبل من اثنين parameter وبترجعلك إما error إما الـ value اللى اتختبرت.

(ex)

```
const ageRule = (value, helper) => {
    if (value == 9) {
        return helper.error('age must be 4');
    أو  return helper.message('age must be 4');
    أو  throw new Error('age must be 4');
    }
    return value;
}
```

② عشان نقدر نستخدم الـ rule custom هنستخدم method تبع الـ joi اسمه custom() بيستقبل الـ rule function.

(ex)

```
const schema1 = joi.object({
    age : joi.number().custom(ageRule),
})
```

https://www.linkedin.com/in/mohamadamir/

الممسوحة ضوئيا بـ CamScanner

(file uploading)

١- عندنا ٣ طرق نقدر نعمل بيها ال uploads :

١- نخزن ال file في ال db على شكل base64 .

٢- " " عندنا في ال server ونخزن ال path بتاعه في ال db.

٣- " " في cloud server ونخزن ال path أو ال url بتاعه في ال db.

٢- عيب ال base64 انه بطيء ويحتاج db كبيرة عشان تستوعب كمية ال file data اللي هتحطها فيها.

أسهل وأحسن طريقة هو إنك تخزن ال file عندك في ال server ولكن محتاج أعمل upgrade لل resource بتاع السيرفر عشان أستوعب لبعد اللي أنا عايزه من ال file data.

ال cloud بيسهل عني عمل تأمين ال data وزيادة ال hard disk.

٣- يفضل نخزن مكان الصورة في ال db، أما الصورة نفسها هتحطها في folder عندنا في السيرفر أو نرفعها على cloud server زي firebase مثلاً أو cloudinary .

٤- ال data العادية بنقدر نبعتها في ال request على شكل json .
أما ال file لازم تنتقل على شكل form-data عشان تقدر تبعت ال binary data زي ال files مع باقي ال data في ال form.

٥- ال html form tag بيحتوي على attribute اسمه enctype وفي بيحدد طريقة تشفير ال data اللي هتنتقل من ال form، نختار "multipart/form-data" لو هنبعت binary data زي ال files.

https://www.linkedin.com/in/mohamadamir/

الممسوحة ضوئيا بـ CamScanner

٦- ازاي بنستقبل الـ string data الـ جاية في الـ request؟
" " الـ binary data " " " " ؟

* في حالة الـ data العادية :
١) الـ data بتتبعت على شكل json format .
٢) الـ nodejs مبيعرفش يقرا الـ data اللي في الـ request لأنها
بتكون متشفرة كـ hexa-codes بالتالي لازم نعمل parsing
الأول ، يعني نحول من hexa لـ string من تاني .

٣) الـ express عنده built-in middleware اسمه express.json()
بيعمل parsing للـ data اللي جاية في الـ req.body .

* في حالة الـ binary data :
١) الـ data زي الصور والـ files بتتبعت في الـ http req على شكل
form-data format .
٢) الـ nodejs مبيعرفش يقرا الـ form-data format بالتالي لازم
نعمل parsing الأول .
٣) هنستخدم 3rd party module اسمه "multer" عشان
نعمل parsing للـ binary data اللي في الـ req.body .

٧- ليه نستخدم الـ multer module؟

١) عشان نعمل parsing للـ form-data اللي جاية في الـ req.body
بالتالي الـ node يقدر يفهمها ويتعامل معاها .
٢) عشان نقدر نخزن الـ files اللي جايلي من الـ req في file system .
٣) عشان نقدر نعمل filteration وتحدد أنواع الـ files اللي مسموح
تتبعت ، وأحجامها كمان يعني .

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

QRCode & Barcode

- ال static files هي فايلات ثابتة مبتتغيرش والـ application بتشغل زي ال file css ، ال js ، ال images وال fonts وال videos وال html .

* ال static files قيمتها ثابتة لكل ال users اللي زي الصور.
* ال dynamic files قيمتها بتتغير / ال o/p بتاعها بيختلف حسب ال input اللي داخلة ، حسب ال users اللي طالبة. زي ملفات ال code backend كال php script مثلاً.

- ال Qr code هي صورة مخزن فيها معلومات.
- ال Qrcode هو الجيل التاني من ال barcode (اللي هو الكود المطبوع بالمنتجات ، من خلاله بنقدر نعرف سعر المنتج)

- ال qrcode اتعمل في اليابان على شكل لعبة لما ، على شكل نقط سودا وبيضا.

- ال qrcode بيقدر يخزن معلومات أكتر من ال barcode.

- باختصار ال qrcode هي صورة بتحمل معلومات معينة.

- إيه استخدامات ال qrcode؟
① بيخزن url لموقع ، وال users بيوصل للموقع دة لما يعمل scan ال qrcode.
② ال payment تحويل الفلوس.
③ بتعمل authentication على جهاز تاني من خلال جهاز عامل auth بالفعل.
④ تخزين معلومات زي معلومات مريض في مستشفى ، معلومات مينيو مطعم ، ...

Task | How to implement Qrcode ...

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

10- الـ cron job هي تاسكات مش بتتنفذ الا
عند وقت معين بنحدده .
أو ممكن نخول هي تاسكات بتشتغل / بتتنفذ تلقائياً
عند ميعاد محدد .

11- الـ cron هي tool بتساعدنا نشغل حاجة عند جدول
زمني محدد ، بتساعدنا نشغل schedual على حاجة معينة .

امثلة : عند يوم كذا ابعت ايميلات لكل الـ users بكذا .
عند يوم كذا الساعة كذا اعمل clear للـ database .
عند يوم كذا الساعة كذا اقفل الـ access على data دية .
زي كورس أو محتوى أو اية كده .

12- لما يبقى عندنا block of code هيشتغل تلقائياً عند وقت
احنا بنحدده ، دة كده احسن " cron job " ، كده إحنا
بنشغل "cron job" .

الـ code ده هي صورة شاملة معلومات .
الـ cron job هي حاجة هتشتغل لوحدها عند وقت محدد .

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

week (8)

1- EJS
2- Ecommerce project
3- hooks
4- Socket.IO
5- Nest 4 TS
6- GraphQL
7- Docker

https://www.linkedin.com/in/mohamadamir/

الممسوحة ضوئيا بـ CamScanner

١- ايه انواع ال pages ؟

عندنا نوعين اساسيين وهما :

① ال static page وده عبارة عن صفحة html بحت

② ال dynamic page وده عبارة عن صفحة html و code بيتحكم

وده بنحققه باستخدام اسم ال template engine.

٢- في حالة ال dynamic page عندنا نوعين من ال rendering :

① SSR وده معناه إن ال code هيشتغل في ال server والناتج

هيتبعت لل browser ، وده بنعمله باستخدام ال template engine

② CSR وده معناه إن ال code هيشتغل في ال browser وده

بنعمله باستخدام ال API .

٣- عندنا مدرستين في ال backend :

① مدرسة ال CSR وده بنبعت API لل client وال client

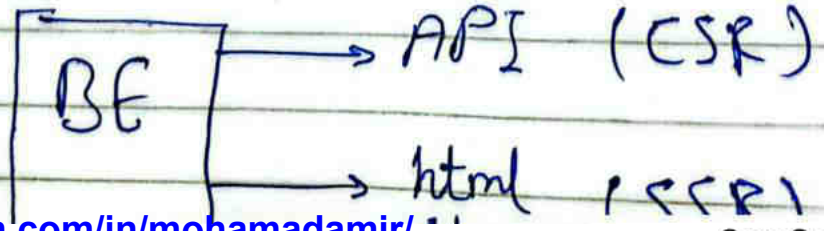
هو المسئول عن عرضه في ال html .

② مدرسة ال SSR وده بنبعت html لل client وال server

هو المسئول عن عرض ال data في ال html .

٤- ال SSR افضل من ناحية ال SEO

ال CSR ~ ~ ~ التفاعلية .



https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

٥- ال template engine هو software بيمكنني من كتابة وتشغيل كود برمجي في ملف html .

تذكر إن ال html هي لغة تصميم وليست لغة برمجة .

٦- في حالة ال CSR ، ال server هيبعت data وال browser هيستلم ال data دية ويعرضها في html باستخدام ال JS .

في حالة ال SSR ، ال server هيبعت ملف html مخطوط فيه ال data ودي باستخدام template engine .

هو في كلتا الحالتين هنستخدم template engine ولكن لو هنستخدمه في ال server يبقى بنعمل SSR
" " " ال browser " " CSR

٧- ال ejs هو اشهر template engine لل JS .
ودي بيسهلنا نكتب كود JS جوا كود html .
ازاي هنستخدمه ؟
① هنعمل folder لازم يكون اسمه "views" .
② هنغير ملفات ال (html) لملفات بامتداد (.ejs) وحطهم جوا "views"
③ هنحط ملفات ال ejs جوا فولدر ال "views" .
④ عشان نبعت ال ejs من ال response هنستخدم res.render()
res.render("index.ejs", { x: "hi", ... });
ال data اللي بايز اعرضها فيه . اسم ملف ال ejs اللي بايز ابعته
⑤ ابدأ اكتب ejs .

<%= x %>
<% %>

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

E-commerce Project

١- فيه خطوات ثابتة بنتبعها لبناء أي مشروع backend :-

١- أول خطوة هي تجميع وتحليل الـ requirements افهم إيه هو المشروع وبيعمل إيه، ودي بتيجي من الـ product owner.

٢- ثاني خطوة هي الـ system design ودي عبارة عن تصميم بنية الـ backend system عبارة عن كتابة الخطوط العريضة
إيه الـ framework اللي هستخدمها
إيه الـ libraries المناسبة
إيه الـ tools المناسبة لبناء هذا الـ system

٣- ثالث خطوة هو تصميم الـ database عبارة عن تجميع الآتي، إيه الـ data اللي هتخزنها.
إيه الـ tables / الـ collection كخطوط عريضة
إيه هي الخصائص في كل table / collection
إيه علاقة كل table بالتاني.
إيه أنسب DBMS هستخدمه.

٤- رابع خطوة بعد تصميم الـ db، صمم الـ API اللي هتتصل بالـ frontend، واتنين بيصمم متحدد: الـ endpoint، نوع الـ req، إيه الـ data اللي هترجع، إيه الـ data اللي هتتبعت، هل فيه auth ولا لا.

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

٥- خامس خطوة بعد ما فهمت ذكره المشروع، وبعد ما
صممت شكل الـ db، وبعد ما صممت الـ APIs، وبعد
ما صممت شكل الـ system (يعني حددت المكونات اللي
هتبني بيها الـ sys (framework، library، tools) ...)

بيجي بقى دور التنفيذ / الـ development وده بينقسم
لخطوتين:

(٥.١) أول setup / جهز الـ development enviroment يعني
أول install و setup للـ framework والـ libraries والـ
tools وجهز الـ configurations المطلوبة للـ database
والـ server ولباقي الـ dependencies.

(٥.٢) ابدأ بقى coding وتنفيذ الـ api والـ business logic
والـ interact مع الـ database.

٦- سادس خطوة الـ security أهم الـ backend
بتاعك من طريق عمل validation لأي input،
أول authentication و authorization وأول تشفير
للـ data المهمة زي الـ password مثلاً.

٧- سابع خطوة أول unit testing عشان تتأكد إن كل
component في المشروع شغال بشكل سليم وتختبر
الـ performance

٨- ثامن خطوة جهز document واضح للكود بتاعك و
للـ APIs عشان الـ developers اللي هيستلموا منك المشروع
يعرفوا كويس إزاي يتعاملوا

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

اشترى ال document لازم تكتب أي معلومة مهمة
زي ال config settings ، ال business logic
APIs ال
، ال deployment instructions ، وأي معلومة هيحتاجها ال developer
اللي بعدك

9 تابع خطوة جهز المشروع بتاعك عشان يتعمله
deploy للـ production environment.

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

٢- مشروع الـ e-commerce أو التجارة الالكترونية هي كل حاجة عايز تبيعها أو تشتريها online وده هو مفهوم الـ ecommerce .

٣- عشان نتعامل مع ecommerce application لازم الأول نفهم الـ requirement بتاعته وده عن طريق ان إحنا نفتح ونبص على مواقع ecommerce مشهورة زي noon ، amazon ، ... وبناء نجمع الـ requirements .

٤- ايه هو الـ slug ؟

- هي عبارة text بينضاف في الـ URL وبيعبر عن محتوى الصفحة . why slug ؟!

١- الـ slug بيخلي الـ user يفهم محتوى الصفحة بمجرد النظر للـ URL .
٢- الـ slug بيحسن الـ SEO .
٣- الـ slug بيخلي الـ URL يبقى readable أكثر ومفهوم أكثر .

- الـ slugify هو عملية تحويل الـ text إلى slug أي تحويله إلى text يمكن وضعه كجزء من الـ URL .

- خطوات تحويل text لـ slug أو خطوات الـ slugify هي كالآتي :
① حول الحروف لـ lowercase .
② احذف أي special character .
③ استبدل أي space بـ hyphen (-) أو بـ underscore (_) .

CS50 Introduction To Web Development
https://www.linkedin.com/in/mohamadamir/

الممسوحة ضوئيا بـ CamScanner

hooks / mongose middleware

١- الـ hooks أو الـ mongose hooks أو الـ mongose middleware
هي حاجة بتتعلق على الـ database .

الـ hooks عبارة عن middleware بيسمحلنا ننفذ logic
معين على الـ data قبل وبعد تنفيذ الـ query .

٢- عندنا two hooks رئيسيين اهم وهم :
① الـ pre hook وهو قبل ما تنفذ الـ query على الـ db اعمل action معين .
② الـ post hook ~ بعد ~ ~ ~ ~ ~ ~ ~ ~

٣- فيه اربع انواع للـ mongose hooks ، النوع اللي هستخدمه
بيتحدد بناءاً على نوع الـ query اللي بتعمله .

يعني مثلاً لو بتعمل insertMany query يعني هستخدم
middleware من النوع (model)

٤- الـ pre والـ post بتعملهم على الـ schemer .
(ex)

```
schemer.pre("remove", function (){
```

بنكتب هنا نوع الـ query اللي عايزين نركب الـ hook عليه

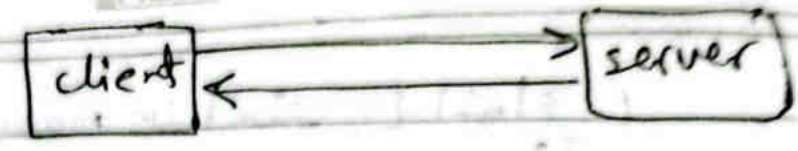
٥- فيه الـ this مختلفة حسب نوع الـ middleware اللي هستخدمه
، الـ middleware هيختلف حسب نوع الـ query اللي هننفذه .

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

٦- الـ middleware بـ mongoose عندنا منه ٤ أنواع ،
نوع بيتعامل مع الـ document والـ document هو الـ data اللي
هتتخزن في الـ db .
نوع بيتعامل مع الـ query وهي أي حاجة ليها علاقة بالـ db
اللي احنا هنعمل فيها الـ data .

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

socket.io



١- ايه هو ال socket ؟ وايه المشكلة اللي حلها ؟

ال socket هي عبارة عن صورة بين ال server وال client تسمح بتبادل ال data بينهم بشكل فوري وتلقائي بدون الحاجة المستمرة لطلب وانتظار action من ال client.

٢- لو عايزين نبعت data سواء من ال server لل client أو العكس، عندنا حاجة اسمها "emitting" يعني ابعت data.

لو عايز استقبل data بنستخدم حاجة اسمها "on"، بنقل بيها listen حال event emit.

٣- ال functions اللي هنستخدمها هي emit و on:
اللي عايز يبعت حاجة بيـ emit والناحية التانية اللي بيستقبل بيـ on

٤- ال socket مينفعش يبقى front بس أو back بس، لازم الاتنين يبقوا شغالين socket.

٥- ال front وال back هيقدروا يكلموا بعض بشكل فوري بدون أي action يحصل / بدون api وذلك باستخدام ال socket.

٦- ال socket هو bidirection يعني رايح جاي، و event-based يعني ال data بتتبعت في event وبعمل عليها listen

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

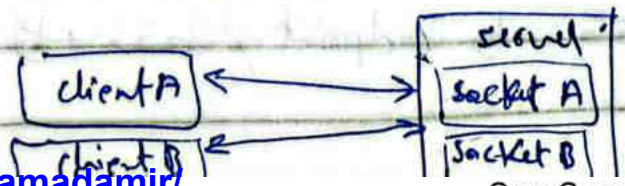
٧- الـ socket مبني على الـ websocket وده عبارة عن communication protocol بيخلي استعمال ثنائي الاتجاه بين الـ server والـ client على عكس الـ http.

٨- الـ websocket بيستخدم في الآتي:
① بنبعت notification.
② نعمل live video زي الـ facebook مثلاً.
③ نعمل call، نعمل comment زي face.
④ نبعت رسايل زي الواتس مثلاً
كل دول بيستخدموا الـ socket لتحقيق ما يسمى بالـ real-time application

٩- امتى نستخدم الـ socket؟
لو عايزين نعمل real-time communication ما بين الـ front والـ back.

١٠- امثلة على real-time communication؟
١- اعمل post فالكل يعرف انك عملت post على face
٢- اعمل comment، like، share فصاحب الـ post يعرف والكل بيشوفه في نفس الوقت.
٣- اعمل live video ...
٤- اعمل رسالة على الواتس أو الماسنجر.

١١- كل client بيبقى ليه socket منفصل لوحده.
وكل socket بيتعامل id unique ولكن في كل مرة بتعمل refresh للـ socket بتاعك بيرجع يعيد من الاول ويعمل id unique جديد.



https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

بديل ال Restfull API — GraphQL

١- قبل ما نتكلم عن ال graph لازم نتكلم الأول عن فكرة ال API، فكرة ال Rest API تتمحور حول ارسال request، السيرفر بـ endpoint معين، وحسب ال endpoint دة السيرفر هيرد عليا بـ data معينة على شكل json.

٢- باختصار ال API هو ارسال req بـ endpoint معين واستقبال data معينة بناءً عليه.

٣- عيب ال API إن كل data مختلفة عايز اجيبها لازم اعمل api بـ endpoint مختلفة، وبكدة هنخلي ال front end يكلم عدد كبير جدا من ال apis ودة هيقلل ال performance وهيقلل ال maintainability

ايه الحل للمشكلة دية؟
الحل هو بدل ما نعمل apis كتير جدا، هنعمل api واحدة فقط وال front-end هو يكلم ال api دية هيقولها هو عايز منها ايه بالضبط ودة هيزود ال performance جدا.

٤- ال GraphQL ليست database وليست لغة لل db زي ال SQL انما هي query language ولكن لل API ال GraphQL بتسمح لك بطلب ما تريده تحديدا باستخدام request واحد فقط.

٥- باستخدام ال GraphQL هتقدر تطلب أي data انت عايزها باستخدام api واحد فقط / endpoint واحد فقط.

https://www.linkedin.com/in/mohamadamir/

الممسوحة ضوئيا بـ CamScanner

٦- في حالة الـ Rest API كان عندنا get و post و put و patch وبس
إنما في حالة الـ graphQL عندنا حاجتين بس وهما query و mutation
لو هتجيب حاجة من الـ db بنستعمل query ،
ولو هتعدل حاجة للـ db ~ mutation .

الـ query هو الـ get ، والـ mutation هي أي حاجة غير الـ get .

٧- في مصطلح الـ query language ، كلمة query تعني سؤال
أو طلب معلومات من الـ db .

٨- في الـ graphQL كل field هو function يعني اقدر
ابعتله parameter .

٩- الـ graphQL بيوفر الـ alias وهو امكانية إعطاء اسم مستعار
للـ data اللي جاية .

١٠- الـ graphQL بيسمح للـ client انه يطلب اي data هو عايزها
باي شكل وترتيب حسب احتياجاته ، باستخدام endpoint واحد .

١١- مش كل الـ data هتكون graphQL ، يعني الـ signup والـ login
والـ logout هتستخدم Rest API عادية جدا .

١٢- بشكل graphQL كل data اللي الـ client يقدر يختار اللي هو
عايزة بس .

١٣- اي حاجة بنبني في الـ graphQL لازم يكون ليها Type .

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

14 أي graphQL object بيكون ليه name و fields
وكل field بيكون ليه type و resolve، الـ type هو نوع
الداتا اللي راجعة، الـ resolve هي الداتا اللي راجعة.

15 الـ graphQL هي client-driven يعني الـ client هو اللي بيحدد الـ api
ويجيب ليه اللي يرجعله.

الـ rest api هي server-driven يعني الـ server هو المتحكم.

في الـ rest api كنا بنعمل endpoints لكل حاجة.

في الـ graphQL منعمل type و schema

في الـ rest بنعمل crud operations
في الـ graph ← query و mutation

في الـ rest بنجيب fixed data بأكثر من api.
في الـ graph ← data محددة بـ api واحدة فقط.

الـ graph أسرع من الـ rest.

الـ graph مبيدعمش الـ file uploading بالتالي منستخدم الـ rest

17 بدأنا نتعامل مع الـ graphQL في الـ nodejs كام
اتنين package يكونوا متوفرين وهما:
① graphql.js وده الـ core الأساسي بتاع الـ graphql
② express-graphql وده middleware بيسهل انشاء
API graphql باستخدام express.

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

١٧- ال graphql هو single api .
ال graphql بيسمح لل front انه يطلب الحاجة اللي هو
عايزها بالضبط من ال back بـ api واحدة فقط بكل مرة .

---

١٨ أي table ، أي schema متعرف object graphql .
أي object graphql بيتكون من name و fields .
أي field بيتكون من type و resolve .
ال type هو نوع ال data اللي راجعة .
ال resolve هو ال data اللي هترجع .
ال resolve هي function بتستقبل arg و parent .
ال arg هي ال argument / ال parameter اللي بنبعتها لل field .
أي field هي function .
دايماً ال data اللي هترجع من ال graphql متعرف راجعة جوا
field اسمه "data" ← { "data": { } }

١٩- ايه مشاكل ال Rest API ؟

١- مشاكل اللي ال front بيواجهها عند التعامل مع ال api هي :
① ال under fetching يعني ال api بيرجعلك data مش كفاية
بالتالي بتحتاج اعمل ال data اللي انا عايزها من كذا endpoint
وأقعد اللف مع بعض ، ودي دايماً بيحصل لو عندي resources
مرتبطين ببعض ، فيه relation بينهم .
② ال over fetching يعني ال api بيرجعلك data كتير وانا محتاج
حاجات معينة منها فقط ، يعني مثلاً انا عايز ال title بتاع
ال product بس ، مترجعليش باقي بيانات ال product انا
مش محتاجها .
③ كل resource مختلف بيعمل ليها endpoint مختلف .

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

Date :

٢١ الفكرة الجوهرية من ال graphql إن بشغل run ل query أو mutation وال front بيطلب الحاجة اللي هو عايزها بالضبط من ال back .

٢١ ال graphiql دي tool ui بيديلك link ال api بتاع ال graphql وهي بترجعلك ال documentation بتاعته ، يعني كده ال graphql بيـ document نفسه بنفسه وده شيء هيخليني كـ front اطلب document من بتاع ال back (:) .

٢٢ ال graphql schema عبارة عن شكل ال api ، شكل ال data اللي ال client يقدر يعمل query عليها . أو يقدر يعمل mutation عليها .

```
import {createHandler} from "graphql-http/lib/use/http";
import {GraphQLSchema} from "graphql";

① const schema = new GraphQLSchema({
    query: new GraphQLObjectType({
        name: "myProjectquery",
        fields:{
            f1:{ type: , resolve: () => }
        }
    }),
    mutation: new GraphQLObjectType({})
});

② app.use("/graphql", createHandler({schema}));
```

https://www.linkedin.com/in/mohamadamir/

الممسوحة ضوئيا بـ CamScanner

Learn & understand nodejs in deep

Don't imitate, understand

لا تقلد، ولكن إفهم

1- Introduction.
2- V8 engine.
3- the node core.
4- modules, exports, require
5- events and the event emitter
6- How nodejs work?
7- Http and being a Server
8- npm
9- express
10- database.

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

① introduction & setup

* مقدمة ، الـ node حاليا" يعتبر أقوى واشهر server technology
بتسهلك تبني موقع ويب كامل بلغة واحدة فقط.

* الـ node واحد من اشهر الـ server technology
اللي بيسهلنا نكتب server code بالـ js.

* عشان تبني كويس في الـ nodejs، لازم تبني كويس في الـ js.

* Q) ايه هو الـ command line interface ؟ ولماذا ؟

* الـ CLI هو برنامج بنكتب فيه اوامر للكمبيوتر عشان ينفذها بدل ما نضغط بالـ mouse على الـ GUI.

* فيه انواع كتيرة الـ CLI وده بيعتمد على نوع الـ OS او نوع الـ computer اللي بنستخدمه.

* Bash on Linux
* Terminal on Mac
* command prompt on windows
* others

Name Tel. Email

* الـ CLI بنستخدم الـ text / الكلمات عشان نؤمر الكمبيوتر.

* الـ GUI بنستخدم الـ mouse.

https://www.linkedin.com/in/mohamadamir/

أغسطس ٢٠١٦
السبت ٦ ١٣ ٢٠ ٢٧
الأحد ٧ ١٤ ٢١ ٢٨
الاثنين ١ ٨ ١٥ ٢٢ ٢٩
الثلاثاء ٢ ٩ ١٦ ٢٣ ٣٠
الأربعاء ٣ ١٠ ١٧ ٢٤ ٣١
الخميس ٤ ١١ ١٨ ٢٥

الممسوحة ضوئيا بـ CamScanner

② V8 the js engine!

* عشان نقدر نفهم الـ node بصورة مناسبة، لازم أولاً نفهم الـ core بتاع الـ node، القلب بتاع الـ node.

* قبل ما نتكلم عن الـ V8 engine لازم نفهم ٣ حاجات الأول وهما: الـ processors، الـ machine code، الـ C++.

1. الـ processor / الـ microprocessor عبارة عن machine صغيرة جداً بتشتغل مع شوية electrical i/ps وفي النهاية بنفذ وظيفة معينة.
2. الـ microprocessor بيفهم instructions/تعليمات بلغة هو بيفهمها.
3. كل microprocessor بيتكلم بلغة معينة، ومش كل الـ microprocessors بيتكلموا بنفس اللغة.
4. لما بنبني instructions للـ microprocessor عشان يشغل برنامج، الـ microprocessor هيستقبل الـ instructions دي بلغة معينة هو بيفهمها.
5. في لغات مختلفة الـ microprocessor اتصمم عشان يفهمها أو يتكلمها، لغات زي: IA-32، X64-64، MIPS، ARM.
6. في لغات مختلفة للـ microprocessor، لكن الـ microprocessor بيقدر يتكلم بلغة واحدة بس.
7. الـ instructions بنقول عليها الـ computer code، وهيا الـ code اللي بنبعته مباشراً على طول للـ machine، بالتالي بنسميها الـ (machine code).
8. الـ machine code

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

(٢) • ال machine code هي عبارة عن لغة برمجة بتتكلم بيل مع ال computer processors.

• كل برنامج بنشغله عالكمبيوتر بيتعمله compile بيعني بيتحول لـ machine code / اللغة اللي ال processor بيفهمها / بيتكلمها.

• كل code بيتحول من النظرية لـ machine code ودة لازم يحصل لأن ال machine من النظرية هي اللي بتعمل كل حاجة.

• المقصود بكلمة machine هو ال processor
• المقصود بال machine code هو ال machine language
• ال machine code صعب جداً جداً نفهمها كبشر، بالتالي عملنا لغات نفهمها كبشر بعد كده من الآخر بنحول اللغة بتاعة ال machine.

(ملحوظة مهمة) • ال node مكتوبة بال C++ والسبب في ده، لأن ال V8 نفسه مكتوب بال C++.

• ال V8 هو الحاجة اللي بتحول الـ code كله لـ machine code، وتعتبر القلب أو ال core بتاع ال node، بالتالي طبيعي ال node يتكتب بنفس اللغة اللي

Name ال V8 engine اتعملت بيها Email

• لغات البرمجة متقسمة من حيث level وكل level بيبعد عن ال machine code بيبقى أبطأ من ال level اللي قبله.

أغسطس ٢٠١٦

السبت		٦	١٣	٢٠	٢٧
الأحد		٧	١٤	٢١	٢٨
الاثنين	١	٨	١٥	٢٢	٢٩
الثلاثاء	٢	٩	١٦	٢٣	٣٠
الأربعاء	٣	١٠	١٧	٢٤	٣١
الخميس	٤	١١	١٨	٢٥	
الجمعة	٥	١٢	١٩	٢٦	

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

Q) ايه الفرق بين ال JS و ال ES؟

في الأساس نفس الحاجة لكن ممكن نعتبر ال JS هي اللغة نفسها و ES هي ال standard اللي بتعرفنا لغة ال JS بتحتوي على ايه وبتشتغل ازاي.

طب ليه فيه standard ظهر للـ JS؟
ليه ال ecmascript ظهرت؟

- لأن كل Browser كان بيجرب ليه engine مختلف عن غيره، يعني كل شركة كانت بتعمل النسخة بتاعتها من ال JS، بالتالي كان لازم يظهر standard الكل يتبعه، وأي engine جديد يتعامل مع JS لازم يتبع ال standard بتاعته واللي هي ال (ES).

- ال ecmascript بيحدد ازاي اللغة بتشتغل، وايه ال features اللي بتملكه.

- أي engine JS لازم يتبع ال ecmascript standard.

Q) ايه هو ال engine V8؟

- هو engine JS خاص بـ google chrome browser.
- هو مكتوب بال C++ و open source.
- بنقدر نستخدمه لوحده، ويمكن ندمجه في C++ application تاني زي ال node مثلاً (ال node هو C++ application).

- ال engine JS هي الحاجة اللي بتشغل ال code JS عن طريق تحويله للغة ال processor بيفهمها.
- ال V8 بيدعم أكتر من لغة processor زي ARB، MIPS، ...

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

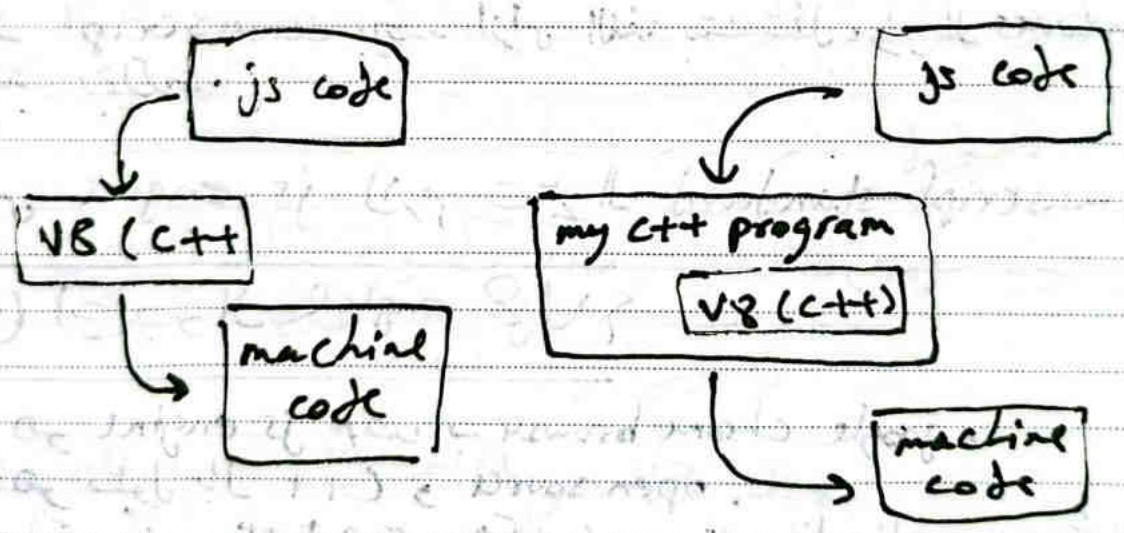
Q) ازاي نقدر نزود features جديدة الـ JS ؟

(1) الـ V8 هو app C++ نقدر نشغله لوحده standalone ، ونقدر ندمجه في app C++ تاني .

(2) الـ V8 بيحتوي على hooks نقدر نستخدمها من الـ app C++ اللي هندمج الـ V8 فيه ، باستخدام الـ hooks دي نقدر نضيف feature للـ JS .

(3) الـ V8 بياخد الـ text JS / JS code وبيحوله لـ machine code .

(4) الـ V8 ما هي إلا C++ program بيستقبل الـ text JS ويحوله compile لـ machine code ، والـ V8 بتسهلنا نكتب C++ code خاص بينا ونخليه متاح للـ JS .



الخلاصة

الـ V8 نقدر ندمجه في أي C++ program ، بالتالي الـ C++ program دي هيقدر يقرأ ويفهم الـ JS ، ومن نقدر نزود feature جديدة من الـ JS غير اللي الـ ecmascript موفراه .

Name Tel. Email

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

Q) ايه فايدة ان إحنا ندمج الـ V8 مع C++ program؟

: الفايدة هو إن إحنا نقدر نزود features جديدة للـ JS.
• الـ JS اتصممت عشان تشتغل في الـ browser،
الـ JS متصممتش عشان تتعامل مع الـ low-level operations
زي التعامل مع الـ db، التعامل مع الـ file sys في الجهاز،
التعامل مع الـ network (لكن) الـ C++ اتصممت عشان
تتعامل مع الـ low-level operations.

• بالتالي نقدر نعمل بالـ C++ حاجة الـ JS مبتقدرش، ونوفرها
للـ code JS كـ feature جديدة من خلال الـ V8 / V8 hooks

V8 engine only = Ecmascript

V8 + C++ program = ES + new feature(s)
standard

• الـ V8 اتصممت بحيث نقدر ندمجها في برامج C++
تانية بالتالي نقدر نوفر feature خاصة بالـ C++ في
الـ JS. ودا هو الـ node.js ☺

• الـ node.js هي C++ program مدمج فيها الـ V8 program
والكلام ده معناه إننا نضيف features جديدة للـ JS خلليها مناسبة
عشان تبقى server technology.

السؤال هنا بقى: ايه الحاجات اللي الـ node ضافها للـ JS؟
وإزاي نستخدم الحاجات دي؟

(ملحوظة) الـ Browser chrom هو C++ program مع V8(C++).

V8 + chrome = ES + (webAPI + DOM + AJAX + ---)
standard

V8 engine

chrom

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

③ The Node Core :-

- احنا قولنا قبل كده إن الـ node هي server technology
هتقدر نكتب server code بالـ JS.

- الـ server باختصار هو جهاز كمبيوتر بيوفر خدمات معينة بنطلبها منه / بنبعت request منه الخدمة.

- الـ client بيطلب الخدمة من الـ server.

- الـ client بيبعت request للـ server، والـ server يرد عليه بـ response مناسب.
الـ response و الـ request بيتبعت بـ standard format معين، والـ standard format دي شرطاً من الـ server والـ client اتصمموا / اتبرمجوا عشان يفهموه.

← الـ process دي بنسميها الـ client-server model
← الـ client و الـ server ماهم إلا computers، واحد بيطلب خدمة والتاني بيقدمها، والطلب والإرسال بيتبعتوا بـ standard format معين وهو الـ http format.

Name Tel. Email

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

Q) ايه اللى كل محتاجاه عشان تعمل/تدير الـ server ؟

Q) ايه الـ features اللى محتاجين نوفرهم من الـ كل عشان نقدر نعمل الحاجات اللى أى web server بيعملها ؟

① الـ كل محتاج طريقة افضل لتنظيم الكود للأجزاء، نقدر نعيد استخدامها، (الـ ES6 حلت المشكلة دية ولكن الـ nodejs فكرت فى الموضوع ده قبل ما الـ ES6 تظهر).

② الـ كل محتاج طرق للتعامل مع الـ files، لأن أى web server لازم هيتعامل مع files (سواء صور، txt، json، ...).

③ الـ كل محتاج طرق للتعامل مع الـ databases، لأن أى web server لازم هيتعامل مع database.

④ الـ كل محتاج القدرة على التواصل عبر الـ internet.

⑤ الـ كل محتاج القدرة على استقبال http requests وارسال http responses.

⑥ الـ كل محتاج طريقة للتعامل مع الـ requests اللى بتاخد وقت طويل بقبل ما تخلص وتتبعت، بالتالى مش مايزين نعطل باقى الـ requests على حساب الـ requests اللى بتستهلك وقت طويل.

← دول عبارة عن شوية مشاكل لازم تتحل لما نعمل أى web server، والمشاكل دية الـ كل مكانتش بتحل لغاية ماظهر الـ NodeJS.

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

Q) ايه اهم اجزاء الـ node core؟

اهم جزء هو الـ c++ core وده المكان اللي الـ nodejs بتتوجد فيه الـ features اللي خلت الـ node تقدر تشغل web servers.

الـ features اللي ماشية من الـ c++ core، بتحتاج من الـ JS من خلال الـ hook اللي بيوفره الـ V8 engine.

Q) ايه أهم folders في الـ nodejs source code؟

اولاً هناك الـ source code بتاع الـ node موجود على github.

الـ source code بيحتوي على كذا folder واهمهم:

① "deps" folder: وده عبارة عن الـ dependencies اللي محطوطة خارج الـ nodejs نفسه، ولكن بنحتاجهم في الـ node، وهما جزء من الـ nodejs. زي مثلاً، الـ V8 والـ libuv والـ npm و...

② "src" folder: وده عبارة عن الـ c++ source code الـ c++ core، وهما ملفات c++ بنبرمج فيها الـ V8 program.

Name Tel. Email

③ "lib" folder: وده عبارة عن الـ core JS أو أكواد الـ JS الجاهزة اللي الـ node عاملها.

أغسطس ٢٠١٦
السبت ٦ ١٣ ٢٠ ٢٧
الأحد ٧ ١٤ ٢١ ٢٨
الاثنين ١ ٨ ١٥ ٢٢ ٢٩
الثلاثاء ٢ ٩ ١٦ ٢٣ ٣٠
الأربعاء ٣ ١٠ ١٧ ٢٤ ٣١
الخميس ٤ ١١ ١٨ ٢٥
الجمعة ٥ ١٢ ١٩ ٢٦

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

Q) هل الـ core c++ هو كل اجزاء الـ nodejs؟

- لأ الـ core c++ هو جزء من الـ nodejs وده الجزء اللي بنعمل فيه features جديدة للـ js زيادة عن الـ features اللي الـ ecmascript مقدماها.

- في جزء تاني من الـ node وهو الـ core js وده الجزء الخاص بالـ libraries الجاهزة أو أكواد الـ js الجاهزة اللي الـ node موفراها عشان تسهل علينا استخدام الـ features اللي عملناها بالـ c++.

- الـ core c++ بنعمل فيه features c++ للـ js.
الـ core js بتسهل علينا استخدام الـ c++ features.

Q) الـ core js موجود فين في الـ node source code؟

موجود في folder اسمه "lib"، والـ folder دي بتحتوي كلها على ملفات .js بنستخدم جواها الـ features c++ عشان تسهل على الـ developer استخدام الـ features دي.

ملحوظة: الـ nodejs بيعتمد على الـ V8 engine الخاص بـ google chrome، بالتالي أي update بيحصل في الـ V8، لازم الـ nodejs تعمل update جديد عشان توفر الـ updates / الـ features الجديدة من الـ V8.

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

تم إضافة أول feature مهمة للكود عشان نقدر نعمل server بيرن، الا وهي طريقة لتنظيم الكود.

④ Modules, Exports, Require

مقدمة: الـ modules والـ exports والـ require يعتبروا واحد من الـ foundational elements في الـ nodejs، والك هتستخدمهم في أي node application هتعملها.

① الـ module عبارة عن reusable block of code ووجودها هيأثر على باقي الكود الأساسي.

الـ كل ملف هتعمله module قبل الـ nodejs وظهر بعد ليل في الـ es6.

الـ module ماهي إلا أكواد نقدر نعيد استخدامها في المشروع ومن غير ما نأثر على باقي الكود.

② الـ nodejs عملت implement لجامع اسمها commonjs modules!!

• الـ commonjs modules عبارة عن standards بتحدد لنا إزاي نرتب الـ modules؟ إزاي نعمل structure الـ modules.

• في الـ nodejs احنا هنبني الـ code كل في module باتباع الـ standard "commonjs".

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

Q) الـ كز هي first-class function، يعني ايه؟

- يعني اي حاجة بنقدر نتعامل مع الـ data زي الـ string والـ number والـ object ... ، نقدر نتعامل مع الـ functions.
١٠

- يعني الـ كز بنقدر نتعامل مع الـ function كـ value
يعني نقدر نخزن الـ f في variable،
ونقدر نبعت الـ f كـ argument لـ f تاني،
ونقدر نعمل return لـ f
١٢

Q) ازاي الـ كز بتسهلنا نعمل function expression؟

لأن الـ كز هي first-class function، يعني الـ function في الـ كز هي first-class يعني نقدر نتعامل معاها الـ value.
٣

(ملحوظة) الـ function في الـ كز هي object، نوع خاص من الـ object.
٥

Q) ايه اهمية الـ modules؟
٦ ① بتخلي الكود بتاعنا يبقى منظم وسهل ادارته.
② بتسهلنا نعمل reusable code منفصل عن الكود الرئيسي.
٧

Q) ازاي نعمل module؟
٨ اولاً الـ module ما هو إلا أكواد كز نقدر نعمل import.

Q) ازاي نعمل import للـ module؟
٩ ← باستخدام function جاهزة تيجي الـ core كز في الـ node
اسمها require()، بتستقبل مكان الـ module كـ argument
وترجع محتوى الـ module أو ترجع قيمة الـ module.exports

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

Q) امتى نستخدم module.exports؟

لما نكون نرجع كود معين في الـ module وليس محتوى الـ module كله.

Q) الـ require() بيجيب ايه؟

بيخلي الـ node يجيب محتوى الـ module كله أو الجزء اللي متعرف في الـ module.exports، إن وجدت.

الـ require() بيستقبل المكان بتاع الـ module اللي عايزين نجيبه.

Q) فين مكان الـ require والـ module.exports؟

موجودين في الـ core js بتاع الـ node.

object & object literal

① الـ name/value pair عبارة عن اسم بيشاور على قيمة.
② الـ js object عبارة عن مجموعة من الـ name/value pairs.
③ في الـ name/value pair الـ value ممكن يكون n/v pair تاني وهكذا...
④ الـ object literal عبارة عن طريقة انشاء الـ object، وهو عبارة عن n/v pairs مفصولين بـ (,) ومتحاوطين في { }.

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

@ ازاى ال node بتعزل الأكواد من ال module عن باقى الأكواد فى المشروع ؟

← لما بنعمل require لأى ملف js ، ال node بيعمله wrapping فى wrapper function مخصوص.

← عموما أى كود js بنبعته فى ال node عشان يـ run ، ال node بيحطه فى function expression معين.

← ال require وال exports وال module بيتبعتوا كـ arguments فى ال wrapper function اللى بنحط فيها الكود ، بالتالى نقدر نعملهم access فى أى ملف js ☺.

(الخلاصة)

① require هى function بنبعتلها 'path' ال module / js file.

② module.exports هى الحاجة اللى ال require function بترجعها.

③ دة بيستعمل لأن الكود بتاعنا بيتلف فى function ، والـ function دى بنبعتلها ال require وال module كـ parameter.

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

٨ ملحوظة عن ال require() function

١ لما بقول انا بـ require لـ module ، يعني انا محتاج منه ن الملف.

٢ ال require بيدور على ملف بامتداد (.js) ، ولو مش لاقية ، اذا هيدور على index.js جوا ال folder ده.

```
(ex) greet
     |-index.js
     |-test.js
```

```
require('./greet');
(is it .js file?
if no then find './greet/index'
```

٣ ال require ذكية بحيث تقدر تفهم إن الملف اللي هيستدعيه مش لازم هيكون (.js) بالتالي ممكن اكتفي باسم الملف من غير الامتداد.

٤ تقدر تعمل require لـ json file ، وال require() هيجي مسؤول عن تحويله ، أي تحويل ال json data اللي فيه لـ js object.

a.json
```
{ "en": "hello",
  "ar": "هلا"
}
```
→ require('a.json') $\xrightarrow{return}$ { en: "hello", ar: "هلا" }

٥ الناتج بتاع ال require بيتعمله caching لكل filename محدد.

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

Q) إيه الفرق بين exports و module.exports؟

① تذكر، لما الكود بيـ run في الـ node، بيتلف من function wrapper معاك كده بتقدر تـ access الـ module obj لأنه بيتبعت كـ parameter في الـ function دي.

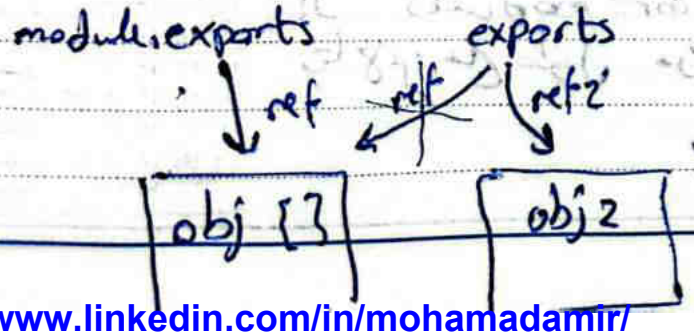
② الـ exports هي اختصار لـ module.exports، والاثنين عبارة عن variables بيشاوروا على نفس الـ object في الـ memory. exports = module.exports;

طب ليه مبنستخدمش exports بدلاً من module.exports؟

ج/ بسبب طبيعة عمل الـ Node. ازاي؟
① لما الـ function wrapper بيتعمل invoke، الـ module.exports بيتخزن في object فاضي، كده الـ module.exports هيشيل الـ reference بتاع الـ object الفاضي ده.

② بعد كده الـ exports بيحط فيه القيمة اللي عايزها. الـ module.exports، بالتالي الاثنين هيشاوروا على نفس الـ object.

③ المشكلة بقى لو إحنا عملنا override على قيمة الـ exports، لأن كده الـ Node هتشيل الـ reference للقيمة الجديدة.
بالتالي الـ exports مش مناسبة لكل الـ module patterns.



https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

@) ايه الحالة اللى نقدر نستخدم فيها ال exports؟

لما نعدل mutate ال exports object، يعني نضيف method أو property ال object من غير override عليه.

بالتالي ال reference مش هيتكسر.

exports = function() { } ✗

exports.m67 = ——— ✓

الخلاصة

استخدم module.exports وخلاص وريح دماغك ☺

ملحوظة: ال version الجديدة من ال V8 من اول ال es6 فيما فوق، بتدعم طريقة لعمل modularity في ال code احسن بكتير من ال commonJS standard.

• الغرض من ظهور ال module هو لاتاحة طريقة لفصل كود ال code وتنظيمه بشكل يمكن إعادة استخدامه

Name Tel. Email

ال core modules قدر نعلمها require بالاسم بتاعها على طول من غير ما نبعت ال path.

(ex) require('utils');

أغسطس ٢٠١٦

السبت	٦	١٣	٢٠	٢٧	
الأحد	٧	١٤	٢١	٢٨	
الاثنين	١	٨	١٥	٢٢	٢٩
الثلاثاء	٢	٩	١٦	٢٣	٣٠
الأربعاء	٣	١٠	١٧	٢٤	٣١
الخميس	٤	١١	١٨	٢٥	
الجمعة	٥	١٢	١٩	٢٦	

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

⑤ event & event emitter

① ايه هو الـ events؟

← الـ event هي حاجة بتحصل في الـ application بتاعنا وبنقدر نـ react / نرد عليها بـ action معين.

← الـ node بيتعامل مع نوعين مختلفين من الـ events.

① اول نوع الـ node بيتعامل معاه هو الـ "system events" ودة بييجي من جزء الـ C++ في الـ node.js core من خلال library اسمها libuv.

الـ system event هو نوع من الـ events بييجي من الـ computer system زي مثلا: الـ file مفتوح، الـ file اتقفل، فيه data وصلت من الـ network، etc... دي الحاجات اللي الـ JS مش قادر يتعامل، لكن الـ C++ قادر.

② تاني نوع هو الـ "custom events" ودي مختلفة تماما عن الـ "sys events".

الـ custom events موجودة في جزء الـ core js في الـ node، ودي عبارة عن events أقدر أعمل create بنفسي والتعامل باستخدام libraries js الـ events، ودة مش جزء من الـ C++، ولكن جزء في الـ core js اسمه "event emitter".

system events	custom events
C++ core	js core
libuv	event emitter

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

(الخلاصة)

الـ node بيتعامل مع نوعين من الـ events :

1) system events وده events بيحصل في الـ system وبيتم ارساله للـ node من خلال الـ libuv.

2) custom events أهنا اللي بنعملها وبنبعتها للـ node من خلال file js اسمه event emitter (events.js)

---

* الـ event هو حاجة بتحصل ونقدر نرد عليها .
* الـ event listener هو الكود اللي بيرد عالـ event .
* الـ custom event هي fake event .
* الـ sys event هي real event بتحصل في الـ sys .

Q) ايه الفرق بين الـ event والـ event listener ؟

- الـ event هو الحدث اللي هيحصل .
- الـ event listener هو الـ code / function اللي هيرد عالـ event لما يحصل .
- الـ event listener هي function بيتعمل invoke لما الـ event يحصل .

- ممكن نعمل كذا / أكثر من event listener على نفس الـ event .

Name Tel. Email

يوليه ٢٠١٦

السبت	٢	٩	١٦	٢٣	٣٠
الأحد	٣	١٠	١٧	٢٤	٣١
الاثنين	٤	١١	١٨	٢٥	
الثلاثاء	٥	١٢	١٩	٢٦	
الأربعاء	٦	١٣	٢٠	٢٧	

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

⑥ How NodeJS works?!!!

في الفصل ده هتكلم عن واحد من اهم الاسباب اللي خلت الـ node تشتغل.

ليه الـ node اتشغلت؟ هتكلم في الفصل ده عن سبب من الاسباب اللي ادت لظهور الـ node

① اولاً asynchronous يعني أكثر من process بيتعملهم execute في نفس الوقت.

② synchronous يعني process واحد فقط بيـ execute في المرة الواحدة.

③ الـ js هي synchronous يعني بتشتغل سطر واحد من الكود في المرة. الـ nodejs هي asynchronous.

④ طالما الـ js هي synch، ليه بنسمع عن الـ js asynch؟!!

اولاً الـ js اتصممت عشان تبقى synchronous، يعني الـ V8 engine اتصمم عشان يـ run الـ js code بشكل synchronous.

لما بنتكلم عن الـ js asynch احنا مبنقصدش الكلام على الـ V8 engine نفسه، ولكن نقصد البرنامج اللي الـ V8 engine مدمج جواه سواء كان الـ browser او الـ node. الـ browser او الـ node بيستغلوا الـ V8 engine مع الـ js في نفس الوقت.

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

⑤ ايه هو ال callback ؟

ال هي عبارة عن function بنبعتها كـ argument لـ function تاني بحيث هيتعامل معاها execute في وقت معين.

function will call/invoke the function that we pass to it back.

ال callback كـ concept هي شوية كود بتحطهم مع شوية كود تاني بحيث يتعامل run عند نقطة معينة والكود التاني شغال في نفس الوقت.

---

⑥ • الـ JS هي synchronous.
• الـ V8 بيـ run الـ code بشكل synchronous.
• الـ libuv هي library c++ اتعملت عشان تهندل الـ events من الـ OS / sys.
• الـ libuv بيـ connects مع الـ OS عن طريق ارسال requests للـ OS.
• الـ libuv بيحتوي على queue بيرتب الـ events اللي خلصت وده بيحصل في نفس الوقت اللي الـ V8 فيه شغال.

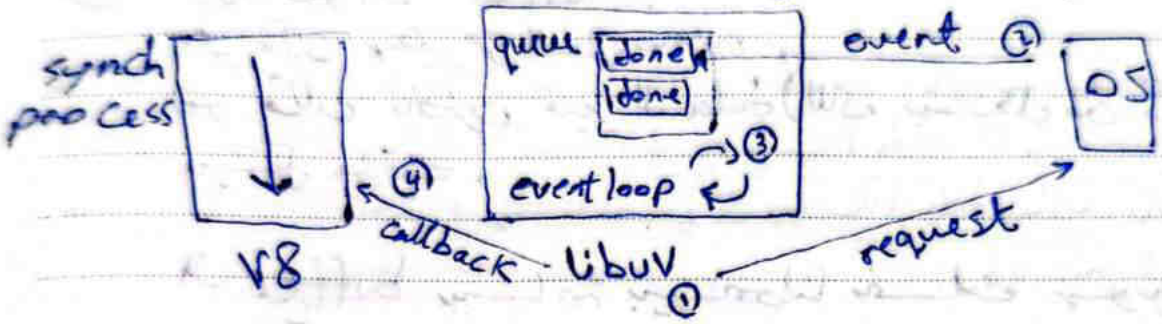
• الـ libuv بيحتوي على اهم جزء وهو الـ event loop وده loop بيمشي على queue of events عشان يـ check لو فيه event خلص ولا لا، وده بيحصل في نفس الوقت اللي V8 فيه شغال.

• اي event بيحصل في الـ sys ما بيخلص بيتحط في الـ queue

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

• كل loop ال event loop بيعمله، هو بيـ check لو ال OS بعت event في ال queue ولا لسه، لو ال event loop لقى event، هيجيبه من ال callback للـ V8 engine عشان يشتغل.

• العملية اللي بتحصل دي هي asynchronous لأنه في أكثر من process بيشتغل في نفس الوقت.



---

(٧) يعني ايه ال node هي event-driven non-blocking I/O in V8؟

← ال V8 هو ال engine js اللي مدمج في ال node عشان يشتغل أكواد ال كل.

← ال I/O دي ال events اللي بتحصل في ال OS زي مثلاً: افتح file، اقفل file، connect بال db، ....

← event driven هو الجزء اللي بنـ request فيه ال event إنه يحصل، ولما يخلص بنستلم notification من ال event loop إنه ال event حصل خلاص.

← non-blocking يعني بنشتغل في حاجة تانية من غير ما نوقف تشغيل البرنامج، يعني أكثر من حاجة بتـ run في نفس الوقت، يعني ال asynchronous

(١١) ايه ال architecture بتاع ال node؟
→ event-driven non-blocking I/O architecture

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

(١٣) ليه اهمية الـ libuv؟

في الـ libuv خلى الـ node تشتغل asynchronously
بالتالي سمحت للـ node انها تتعامل أكثر من
process في نفس الوقت، بالتالي حسنت الـ performance.

(١٤) ليه هو الـ buffer؟
هو مكان صغير في الـ memory.
هو مكان بتخزن فيه الـ data (اللي بتنتقل من مكان لتاني)
بشكل مؤقت.

الـ buffer بساطة بيعملنا نفس شوية data
بعد كده تنتظر، امسك وانقل، امسك وانقل.

(١٥) ليه هو الـ stream؟

هي سلسلة من الـ data اللي متوفرة مع مرور الزمن.
هي زي نهر من الـ data ماشي طول الوقت.
هي مجموعة من الـ data بتنتقل من مكان لمكان تاني
مع مرور الزمن، وبنعمل processing للـ data وهي بتنتقل.

مثال: لما بنشوف فيديو على اليوتيوب، لما بنعمل download
لفيديو وتتفرج عليه / بتشغله في نفس الوقت.

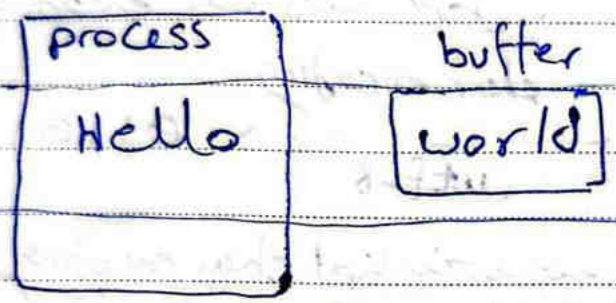
https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

ازاي بنجمع بين الـ stream والـ buffer

الـ stream هي سلسلة من الـ data بتتنقل مع الوقت وبيتعامل معاها process اثناء وهي بتتنقل.

الـ buffer هي ذاكرة مؤقتة بنحطها فيها الـ data اللي بتتنقل من مكان للتاني.

١) الـ data بتتيجي من الـ stream وبتكون كتيرة جداً.
٢) مجموعة من الـ data اللي جاية هتتمسك في الـ buffer.
وقبل ما الـ data اللي بيتعامل process تتخلص والـ data الباقية توصل.



٣) الـ process بيخلص لما الـ stream يتشال.

Q) ايه هو الـ error-first callback

هي مجرد function بتستقبل error object كأول parameter وبتبقى by default، وباقي الـ parameters هي callbacks جاية.

لو معنيش error اذا قيمة الـ parameter هتبقى null، وبعد كده هيكمل run مع باقي الـ parameters، وإلا هتنقل object يتعرف الـ error ده.

الـ pattern ده بيساعدنا نرتب الـ callback functions.

```
function (err, data){

}
```

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

Q) ايه هو ال character sets ؟

- هو عملية تمثيل الحروف على هيئة ارقام.
- كل حرف بيبقى ليه رقم معين.
- ال unicode و ال ascii هما أهم انواع ال character sets.

Q) ايه هو ال character encoding ؟

- هو الطريقة اللى هنخزن بيها الحروف فى binary code.
- كل حرف بيبقى ليه رقم على حسب ال char set اللى بنستخدمه، وكل رقم هيتحول ويتخزن فى binary code حسب طريقة ال encoding اللى هنستخدمها.

$$h \xrightarrow[\text{unicode}]{\text{char set}} 104 \xrightarrow[\text{utf-8}]{\text{char encoding}} 01101000$$

- فى ال char encoding احنا بنحدد عدد ال digits اللى هتمثل بيها الحروف فى binary code، وال utf-8 بتستخدم 8-digits بالتالى بتقدر تمثل عدد كبير جدا من الحروف، عشان كده ال UTF-8 اشهر نوع بيتم استخدامه.

Q) هل ال كل / ال V8 engine يقدر يتعامل مع ال char encoding ؟

ال V8 يقدر يتعامل مع ال char sets اى ال unicode لكن معندوش mechanism للتعامل مع ال encoding، ال node وفرت طريقة للكل للتعامل مع ال binary data.

Name Tel. Email

يوليه ٢٠١٦

السبت	٢	٩	١٦	٢٣	٣٠
الأحد	٣	١٠	١٧	٢٤	٣١
الاثنين	٤	١١	١٨	٢٥	
الثلاثاء	٥	١٢	١٩	٢٦	
الأربعاء	٦	١٣	٢٠	٢٧	
الخميس	٧	١٤	٢١	٢٨	
الجمعة	١	٨	١٥	٢٢	٢٩

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

Q) ازاي الـ node بتتعامل مع الـ stream ؟
اية هو الـ chunk ؟

مع الـ stream ماهو إلا سلسلة من الـ data
بتنتقل مع مرور الوقت، الـ data اللي بتتبعت
من الـ stream بنقول عليها chunk.

← الـ chunk هي الـ data الواصلة اللي بتنتقل من الـ stream.

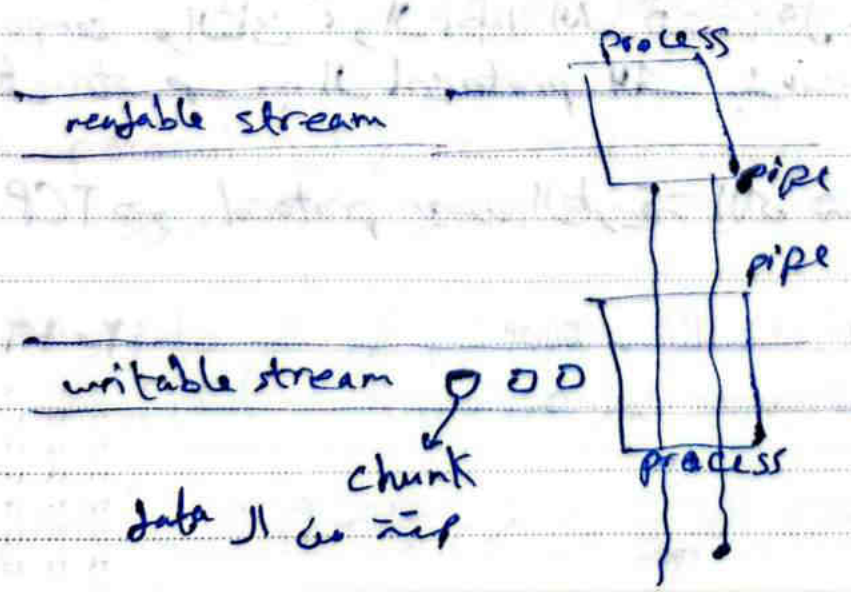
مع في الـ core js فيه ملف اسمه stream.js بنستخدمه
للتعامل مع الـ stream بالـ js.

Q) اية هو الـ abstract class ؟

← هو class مبنتعملش عليه بشكل مباشر ولكن
بنورث منه.

Q) اية هو الـ pipes ؟

هو link أو connection بيربط بين 2-streams
شغالين، stream بيعمل write والتاني بيعمل read.



https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

(7) HTTP & being A server

① إيه هو الـ protocol ؟ TCP/IP

← الـ protocol هي سلسلة rules بتتبعها وبتتفق عليها أثناء التواصل.

← الـ client والـ server اتبرمجوا عشان يفهموا الـ rules دي ويستخدموها أثناء الاتصال / التواصل ما بينهم.

← فاكر الـ client-server model وهو إزاي الـ server والـ client بيكلموا بعض.

← كل computer بيبقى ليه IP خاص بيه، الـ IP هو الرقم اللي هنتعرف بيه الـ computer اللي عايزينه تكلمه.

← لما بنعمل connection من computer لـ computer تاني، الـ network OS هيفتح socket ويـ connect على الـ computer التاني.

← الـ sockets هو الـ link اللي الـ info بتنتقل ما بين computer والتاني، والـ info اللي هتنتقل بيتعمل structuring حسب الـ protocol اللي بنتبعه (ex: http)

← الـ TCP هو protocol يحدد الطريقة اللي هتتبعت فيها الـ info.

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

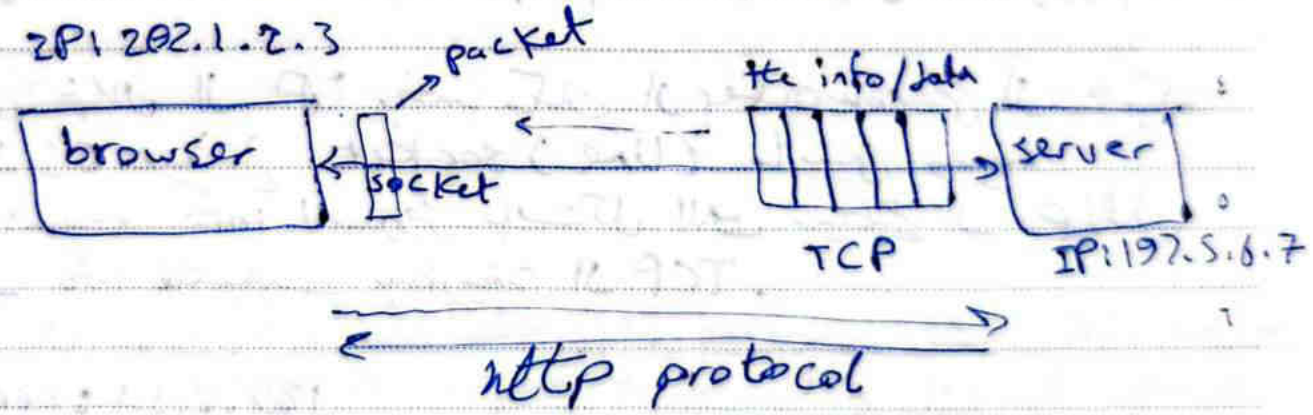
@ ازاى نوصل لكمبيوتر معين؟ من خلال ال IP address بتاعته.

@ ازاى بنتصل connect مابينهم؟ ال OS هيخلى ال network بيفتح socket منتقل فيه ال info.

@ ازاى هنبعت ال info؟ من خلال protocol مناسب يحدد ازاى هنبعت ال info؟ وهنبعته على شكل ايه؟

@ ازاى ال TCP بيشتغل؟ ١) بيقسم ال info اللى هنبعتها لقطع وكل قطعة منهم هتتبعت لوحدها من خلال ال socket فى المرة الواحدة.

القطعة دى بنسميها "packet"



(ملحوظة) ال information ذات نفس ليها protocol معين متشكلة عليه سواء Http protocol أو غيره.

اما الطريقة اللى هتتقسم فيها ال info والطريقة اللى هتتبعت بيها هى دى ال TCP

ال OS عنده القدرة على فهم واستخدام ال protocols دى.

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

Q) (ع) أنواع الـ server
① web server
② file server
③ email server

الفكرة بتاعة الـ TCP شبة الـ stream اللى الـ node بتستخدمه.

← HTTP ← protocol بيحدد شكل الـ data هتتبعت ازاى.
← IP ← بيحدد مكان الـ computer فين فى الشبكة عشان نقدر نوصله..

← TCP ← protocol بيحدد طريقة ارسال الـ data عبر الـ socket.

Q) ايه الخطوات اللى بتتم عشان نوصل للسيرفر؟

1- من خلال الـ IP بنحدد مكان الـ machine فى الشبكة
2- الـ OS هيفتح socket (مدخل) مابين وبينه.
3- هنبعت info للسيرفر بالشكل اللى بيتطلبه الـ http.
4- الـ info هتتبعت بطريقة الـ TCP.

129.2.3.1 : 220
IP address port

الـ IP address هو عنوان الجهاز اللى مابين نوصله فى الشبكة.
الـ port هو مكان البرنامج اللى مابين نوصله من الجهاز.

Name Tel. Email

الـ http method بيحدد نوع الـ action اللى الـ request مابين عايزه.

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

Q) ايه هو الـ HTTP؟

- هي مجموعة rules و format لـ data اللي هتنتقل عبر الـ TCP/IP / الـ internet / الـ web.

- الـ HTTP هو الـ protocol اللي هو للتواصل مع الـ web server.

Q) ايه هو الـ http-parser؟

هو عبارة عن برنامج مكتوب بلغة الـ C، بيسمحلنا نـ parse، نحلل الـ HTTP request والـ HTTP response بحيث نقدر نفهمهم ونتعامل معاهم في الكود.

Q) ازاي نحمل HTTP request و HTTP response بالـ JS؟

من خلال file JS موجود في الـ core JS في الـ node باسم "http.js".

- احنا نقدر نخلي server يعمل listen لـ data اللي المفروض تيجي للسيرفر، والسيرفر هيبعتها لـ port اللي الـ node لازم يعمل listen عليه.

Q) ايه هو الـ routing؟

هو عملية mapping الـ HTTP requests لـ content، والـ content دة ممكن يكون file في السيرفر او لا.

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

(8) NPM

① ايه هو ال package؟

ما هو إلا code ، library ، module ، مجموعة أكواد
اقدر استخدمها في المشروع ، واقدر اعمل عليها
management باستخدام package management system .

الكود اللي بخطه في system مش قادر نديره فيه ،
بنقول ما الكود ده package .

② ايه هو ال package management sys؟

هو software بيعمل install و update للـ packages
من خلال ال system اللي بتحتفظ فيه بال packages .

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

⑨ Express

الـ express هي node package بتسهل علينا عملية انشاء الـ web Server ,

الـ express خلت كتابة الـ server code اسهل ، مما كانت عليه لما كنا بنكتبها بالـ core node .

ايه هو الـ environment variables ؟

هي عبارة عن global variables الـ environment اللي الكود بتاعنا موجود فيها ، من الحالة بتاعتنا ، الـ environment هو الـ process .

القيم بتاعة الـ env variables بتكون قيم خاصة بالـ settings والـ configurations الخاصة بالـ environment .

process.env.PORT → variable الـ اللي عايز اجيب قيمته

الملف اللي فيه الـ env variables

ايه هو الـ routing ؟

هو عملية ربط url معين بـ content معين سواء الـ content ده text ، html ، صورة ، json ، ...

في الـ route احنا بنعمل mapping ما بين الـ url والـ content .

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

ايه مكونات ال url؟

url pattern = 1) http method
2) http protocol
3) base url
4) endpoint

• كل endpoint بيمثل url واحد.
• كل url ممكن يمثل أكثر من action، get، post، ...

ايه هو ال middleware؟ ال gate / البوابة

ال middleware هو كود أو software بيتحط ما بين two layers of software.

في حالة ال express، ال middleware بيتحط ما بين ال request وال response، ما بين ال client وال server.

ايه هي ال static files؟

هي ال files اللي مش dynamic مع ال server code يعني ال files اللي مش بيتعمل process أو generate من ال server code.
زي مثلاً، html، css، js، image، ...
بتبعت لل client زي ما هي.

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

Q: ايه هو ال cookie-parser؟

هو middleware جاهز تبع ال express، بيستخرج ال cookie من ال request ويحطها في property اسمها cookies تبع ال express request object.

→ req.cookies

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

** (Nest js) notes

١- ايه هو ال TypeScript ؟

هي development tool مبنية على ال javascript بتساعدك تكتب كود js بشكل جيد من غير loosely type، يعني بتخلي ال js تتصرف كأنها strongly typed language وب syntax جديد أقرب للغات ال OOP.

٢- ايه ال pros او ايه المميزات اللي ال TS هتضيفها لل JS؟

١) خلي ال js تتصرف كـ strongly typed language بالتالي حل مشكلة ال loosely type في ال js

٢) اضافة datatypes جديدة لل js زي مثلاً ال interface وال enum

٣) خلي ال syntax بتاع ال js يكون أقرب للغات البرمجية العادية اللي بتدعم ال OOP كـ class-based

٤) بتخلي أكتشف الأخطاء البرمجية أثناء ال development

٣- ايه هو ال nestjs ؟

هو nodejs framework مبني على ال express وهنا ال framework وفر opinionated structure وأيضاً الناتج ال tools وال libraries اللي هتحتاجها لبناء أي مشروع backend بال nodejs.

https://www.linkedin.com/in/mohamadamir/

الممسوحة ضوئيا بـ CamScanner

٤- ايه هو ال @nestjs/cli ؟

هو اسم الشخصي لل nest ،
هو ال tool اللي من خلاله هنقدر نعمل إنشاء
لمشروع nest ونتعامل مع الملفات والمجلدات
اللي فيه .

```
>> npm i -g @nestjs/cli
>> nest new blf
```

اول ما بكتب كلمة nest في ال cmd ، يعني
انا بكلم ال cli nest ، يعني بشغل ال cli nest

٥- ايه الفرق بين ال npm وال yarn ؟

اولاً ال package manager هو مدير ادارة المكتبات
واحد العمل اللي تخص لغة معينة ، كل لغة برمجية
لها pm خاص بيها سواء (php ، py ، js ، java ...)

ثانياً وهو الفرق :
* وانا بنزل package بال npm وحصل انقطاع لل شبكة
الانترنت ، لازم هنشغل كل اللي عملناه ونبدأ من
أول وجديد لما الاتصال يرجع .
اما ال yarn بيكمل تنزيل من نفس النقطة
اللي هو وقف عندها .

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

NO.: رقم :
Date : التاريخ :

٦- اي حاجة بايزين نعملها من ال nest، هنعملها بال nest/cli (الأمر الشخصي بتاع ال nest)

٧- ١- شرحلي هيكل مشروع ال nest؟

① مجلد ال node_modules اللي بيحتوي على ال source code لجميع المكتبات اللي معمولها install.

② مجلد ال src وفي ال base component اللي بيحتوي على أكواد ال backend.

③ مجلد ال test الخاص بال unit testing

④ ملف ال gitignore اللي بيحتوي على أسماء الملفات اللي مش هيتعمل push لل github.

⑤ ملف ال package.json اللي بيحتوي على معلومات عن المشروع وجميع المكتبات اللي تم انزالها بال npm.

⑥ ملف ال tsconfig.json للتحكم في إعدادات ال ts

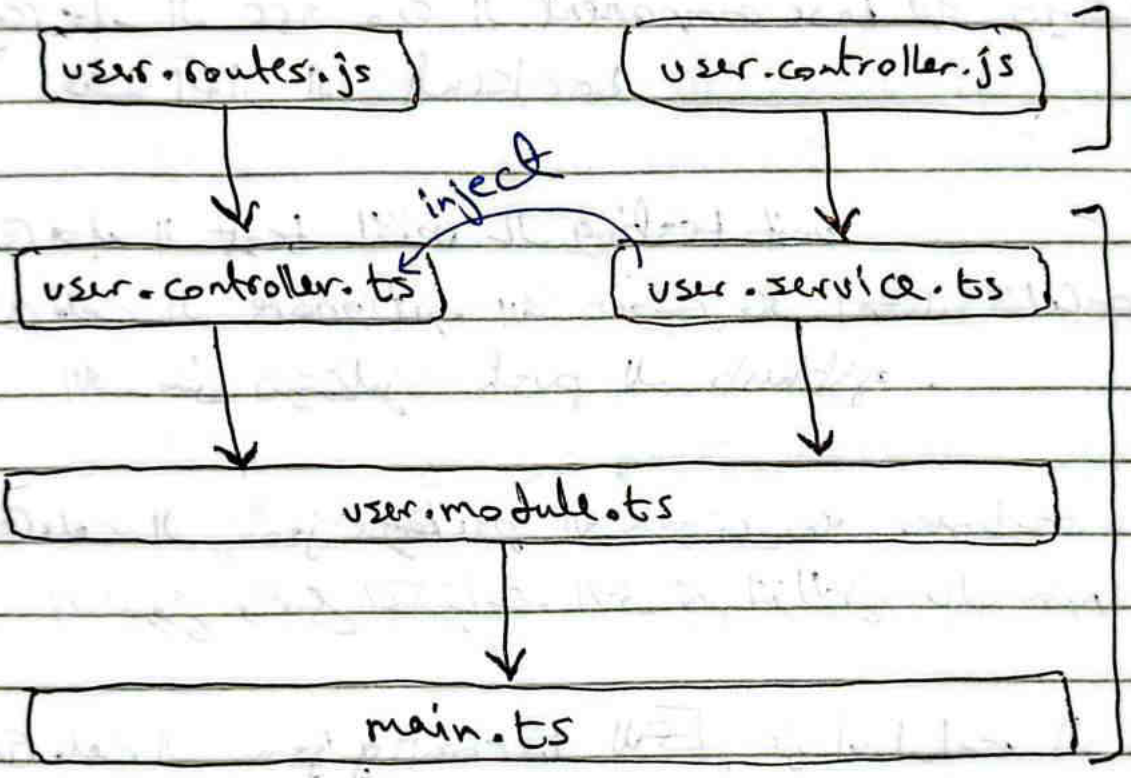
⑦ ملف ال nest-cli.json التحكم في إعدادات ال cli

شغلنا كله هيكون في مجلد ال src.
كل متر كنت هيكون على ال src

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

٨- احنا من ال express كنا بنشتغل على
ملفين اساسيين :
① ملف routes ← user.routes.js
② ملف controller ← user.controller.js

من ال nest js ملف ال routes بقى controller
وملف ال controller بقى service وبال ts



٩- أي controller و service منفصلة لازم يكون
ليهم إنتماء لـ module ، لازم يكونوا تبع module
(الـ module ماهو إلا مجموعة من ال services والـ controllers)

١٠- علشان ال module يـ run ويشتغل ، لازم ينتقى
ملف ال main.ts ، وده ال entry point اللي
بيشغل ال server يعني هو اللي بيكوم الدنيا .

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

NO.: Date :

11 الخلاصة من رقم ٩ و ١٠ :
عندنا ملف main.ts بيربط الـ modules ، والـ module
بيربط الـ services و controllers ، وملفات نستخدم
الـ service جوا controller لازم نعمل inject جوا

12- أي built-in method بيبدأ بعلامة @ دة بنسميه
decorator وكل وظيفته إنه يضيف features
زيادة للـ class أو للـ method اللي مكتوب بعديه
بحيث يحولهم من class عادي أو method عادي
إلى class ليه اختصاص معين في الـ nest أو
إلى method ليه ... ليه ... ليه
مثلاً :
* الـ nest مش هيتعامل مع الـ class على إنه controller
إلا لما أعمل call لـ @Controller() قبل الـ class .

* الـ nest مش هيتعامل مع الـ class على إنه module
إلا لما أكتب @module() قبله .

* الـ nest مش هيتعامل مع الـ class كـ service
إلا لما أكتب @Injectable() قبله .

لو مكتبناش decorator ، الـ nest مش هيفهم
الـ class أو الـ method دي بيعمل إيه وهيتعامله
كأنه class عادي زي أي class .

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

NO.:
Date :

١٢ ال nest موجود جواه built-in error handling يعني
مش محتاج اعمله بنفسي زي ما كنا بنعمل في ال express
ودة من مميزات ال nest انه بيعمل عني هم معالجة
ال errors اللي هتظهر في ال response .

nest Vs express
من مميزات ال nest

١٤ أي تعديل بنعمله في أي ملف ts في ال nest لازم
نعمل restart للسيرفر علشان يعمل recompile
لملفات ال ts بالتالي التغيير يبقى معندي .

افضل حل هو إني اهنا نستخدم ال watch feature
في ال nest ودة هيخلي السيرفر يعمل restart
بشكل أوتوماتيك لو حصل أي تغيير .

```
>> nest start --watch
```

١٥- اشرح ال code التالي :

```
@Get('/blz')
getblz(){
    return "hello"
}
```

لو السيرفر استلم request من نوع get و
ال path بتاع ال url كان /blz ، في الحالة دي
اعمل invoke لل getblz() .

١٦- في ال post request وال put request اهنا بنبعت
json data في ال body ، ولازم نحول ال json
اللي جاي لـ object في عادي علشان نقدر نستخدمه
في كود ال (ودة بنقول عليه json parsing)

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

NO.: Date:

١٧- من ال express بستخدم middlewares احسن
express.json() ساعات نحل ال parsing ال
ال جاي من ال post/put request وتخلي ال server
يقدر يحلل ال json data دة بشكل صحيح.

بدون ال express.json() مش هنقدر نستخدم ال
ال جاي من ال post request بالتالي ال req.body
هيكون ب undefined.

١٨- ال nest جاي معاه ال json parsing مدمج جواه
بالتالي مش لازم اعمله بنفس زي ما كنا بنعمل
من ال express.
يعني ال json ال جاي من ال body بتاع ال post req
هيتم تحويله من buffer لـ js object بدون تدخل
مني.

nest vs express
nest ال

١٩- (لاحظ) ال data ال بتكون من ال body بتاع ال request
بتكون على شكل buffer data بالتالي
لازم نعمل parsing يعني تحويل من
buffer لـ json ثم إلى js object
ودة باستخدام express.json()

٢٠- نقدر نستقبل ال data ال مبعوتة من ال body باستخدام
ال decorators @Body() ، @Post('/')
addProfile(@Body reqbody:any) {
console.log(reqbody);
}

الخام ال من ال body
هيتعمل parsing
وهتستخدمه هنا

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

NO.: Date:

21- ازاى نقرأ ال parameter من ال URL في ال request؟

```
@Post('/:id')
addUser(@Body b:any, @Param() p:any, @Query() q:any){
    console.log(b)
                (p)
                (q)
}
```

22- ال return في ال nest بيبعت response تلقائيا من غير ما تكتب res.json أو res.send وبيبعت معاك status(200).

```
return
=
res.status(200).json()
```

nest vs express

من ملاحظة الـ nest

23- ازاى نعمل custom response في ال nest؟

```
@Get('')
getData(@Res() res:Response){ }
~~get~~
@Get('')
getData(@Req() req:any, @Res() res:any){ }
```

ال @Res() وال @Req() هيخلوك أقدر استخدم ال req object وال res object من مكتبة express واستخدمهم زي ما بتستخدمهم في express

https://www.linkedin.com/in/mohamadamir/

الممسوحة ضوئيا بـ CamScanner

NO.: Date :

٢٤- ايه اهمية الـ decorator @Injectable() ؟

بيعرف الـ next إن الـ class اللي مكتوب جنبه أقدر أعمله نسخة منه من class تاني بطريقة الـ dependency injection design pattern ، والـ next هو المسؤول عن تنفيذ الـ design pattern دي من وراء الكواليس

٢٥- ايه فكرة الـ dependency injection ؟

لو عندي class A و class B ، باستخدام الـ DI أقدر أعمل نسخة من الـ class A جوا الـ constructor بتاع الـ class B من غير ما تحتاج الـ keyword new بالتالي لو في المستقبل قمنا بتعديل الـ constructor الخاص بالـ class A مش هنضطر نعدل على النسخ اللي متعملة في الـ classes التانية ودي لأن الـ constructor new بتتم من برة الـ class أصلاً وبنجيبها من خلال الـ constructor بتاعه.

وبكدة إحنا قللنا الاعتمادية بين الـ classes ، لأن مثلاً في الـ class A مش هيعرف معايا لدين الطريقة دي مش بتناديه الـ constructor بشكل مباشر.

٢٦- الـ DI بيخلينا أعمل instance من كلاس جوا كلاس تاني من غير ما نحتاج الـ keyword new بشكل مباشر ولكن برة برة.

https://www.linkedin.com/in/mohamadamir/

الممسوحة ضوئيا بـ CamScanner

NO.:
Date :

٢٧- ايه هو الـ dto ؟

الـ data transfere object هي schema بتحدد بيه شكل الـ data اللي هتستقبلها من الـ request body

كـ implementation الـ dto هو class بنكتب فيه اسماء الـ properties اللي لازم تكون موجودة في الـ req body وتحدد الـ datatype بتاع كل property وممكن نضيف validation معين لكل property بحيث في النهاية هنستخدمه على الـ @Body() b علشان نتأكد هل الـ user بعت الـ data صحيحة ولا لا ؟

اي data متوقع إن احنا نستقبلها من الـ post req أو الـ req كلها هنعمل ملف dto خاص بيها

نقوم بإنشاء ملف dto لضمن الـ validation

٢٨- ازاي نعمل validation من الـ nest ؟

اولاً : npm i --save class-validator class-transformer

ثانياً : علشان الـ validation يشتغل لازم نروح لملف الـ main.ts ونضيف جوا الـ bootstrap app.useGlobalPipes(new ValidationPipe());
بالتالي الـ validation هيشتغل على مستوى الـ application كله

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

NO.: Date :

١٩- ال pipe هو function بياخد من input وكلاس الأقل
ويعمل عملية تعديل معين بعد كده يرجعهولك متعدل.

٢٠- ازاي نعمل generate ل CRUD بال cli؟

>> nest g resource todos

* الأمر ده هيعمل إنشاء ل
ملف module و controller فيه كل ال CRUD
وملف service و dto و entity

* شبه أمر انشاء app جديد في Django.

٢١- أي module جديد هنعمله لازم لازم نضيفه
في ال import بتاع ال appModule بتاعك ال
main.ts بتوقف وبتشتغل. وإلا هيتم
اعتبار إن مش موجود أصلاً.

* شبه لما نضيف app في ملف ال settings.py
في Django.

٢٢- ليه يفضل نعمل ال dto كـ class وليس interface؟

لأن ال interface في ال ts ملهوش مقابل
في ال js، ولكن ال class ليه مقابل في ال js.

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

٢٣- ازاى نتعامل مع ال db من ال Nest؟

١- ال TypeORM هى عبارة عن مكتبة بتسهل التعامل مع قواعد البيانات، وهذه المكتبة تندرج تحت نوع من ال tools اللى اسمها ال ORM.

٢- الفكرة الاساسية وراء مكتبات ال ORM انها بتحول الجداول من قواعد البيانات الى objects. وبالعكس بالتالى اقدر اتعامل مع ال database على إنها object.

٣- ال TypeORM بيشتغل على ال nodejs ومكتوبة بال TS.

٤- ال TypeORM بيدعم مختلف ال sql databases زى: sqlite و MySQL و MariaDB و Oracle و MS SQL Server و PostgreSQL.

٥- ايه مميزات ال TypeORM؟

① بيسهل ال connect مع اى relational db. مجهولة طالما بيدعمها.

② بيسهل انشاء الجدول من ال db بناءا على class ts انا كاتبه، وهذا ال class بنسميه بال model او ال entity.

③ بيوفر methods جاهزة لل DML بدون كتابة اى sql code.

④ بيسهل عمل relationship بين الجداول

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

NO.:
Date :
رقم :
التاريخ :

٢٤- ما هو ال entity ؟

ال entity هو عبارة عن ال object الاساسي اللي بيمثل ال table من قواعد البيانات لما كنا نتعامل مع ال TypeORM .

كل جدول من قواعد البيانات هتمثله entity خاص بيه ، وال entity ما هو إلا class يمثل جدول من ال db .

عملية تحويل ال entity إلى جدول في ال db يتم من خلال ال TypeORM .

٢٥- ايه هو ال @Entity() ؟

ده decorator لازم نحطه قبل ال entity . بالشكل ال nest يفهم إن هذا ال class هو entity ولازم يتعامل mapping بحيث يظهر كـ table في ال db اللي متصل بيها .

افتح ال documentation بتاع ال nest وروح لـ
① Configurations بالشكل تعرف ازاي تستخدم ال environment variables في ال nest

② Database بالشكل تعرف ازاي تتصل db باستخدام ال TypeORM

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

٢٦. ايه هو الـ guard؟

هو class متعلم بـ Injectable decorator وبيعمل implements لـ CanActivate interface يعني لازم يحتوي على method اسمها CanActivate بترجع إما true أو false.

الـ guard بيحط قبل route معين عاشان يعترض الـ req اللي جاية، لو رجع true هيكمل الـ req، لو false هيمنع الـ req من انه يكمل.

بنستخدم الـ guard عشان نحمي الـ resources بتاعتنا اللي هي الـ authorization.

٢٧- أي auth system متكامل في أي backend هيكون كالآتي:
أولاً نجهز منطق برمجي لـ signup عشان نسمح للـ client انه يعمل حساب عندي عشان يقدر يوصل لـ resources مخصصة للـ client عندي فقط.

ثانياً جهز منطق برمجي لـ signin عشان تتأكد إن الـ client عندي حساب عندك فعلاً.

رابعاً: هنعمل jwt لكل عملية signin عشان تخلي الـ backend يعرف فاكر الـ user اللي عمل login طالما بيبعت الـ token مع كل req بالتالي مش هيضطر يعمل login مع كل req بيبعتها، إنما هيعمل login مرة واحدة طالما مش عاوز logout (مسح الـ token) وطالما صلاحية الـ token لسه مش expired

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

NO.: Date:

٢٨- ايه ال flow بتاع ال authentication؟

اولاً ال user هيبعت post req فيه ال email وال pass
ثانياً ال backend app هيستقبل ال req ويتأكد هل ال user
مسجل عندي، يعني هل هو user valid؟

ثالثاً لو user valid هيتم عمل generate لل token الخاص
بال user.

رابعاً ال backend هيبعت ال token كـ res لل client
خامساً لو ال client عايز يعمل access لل endpoint محمي، إنما
لازم يبعت ال token من ال header بتاع ال req

سادساً ال backend هيتأكد اذا كان ال token صحيح
ولا لأ.

سابعاً لو ال token كان valid اذا ال backend هيرد
عليه بال res لل req اللي بعته، وإلا
ال req هيترفض.

٢٩- ما الغرض الاساسي لل auth و ال authz؟

حماية ال resources

٣٠- بتقدر تعمل api documentation بشكل احترافي
باستخدام مكتبة swager والشرح موجود
في ال rest documentation.

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

(software testing)

(مقدمة)

هذا الشرح سينقسم لـ 7 اجزاء :

① في الجزء الاول هنشرح المبادئ والـ concepts الاساسية
② " " الثاني " الـ unit testing .
③ " " الثالث " الـ integration testing
④ " " الرابع " الـ E2E testing
⑤ " " الخامس " الـ TDD والـ BDD .
⑥ " " السادس " الـ performance testing
⑦ " " السابع " الـ Jest framework

① Concepts :

*** دة اهم سيكشن . لانه هنتعرف فيه على الاساسيات العامة للـ software testing وتعرف الفرق بين الـ manual testing والـ automation وتعرف المميزات اللي هناخدها من الـ automated testing وتعرف انواع الـ testing المختلفة والـ tools اللي هتساعدك علشان اقدر اعمل testing للـ application بتاعي .

1- ايه هو الـ software testing ؟

- هي طريقة بنختبر بيها اداء الـ application اللي عملناه وبنتأكد بيها ان كل الـ features شغالة بدون اي وقف وبتطلع الناتج اللي متوقع منها .
- علشان نقدر نعمل testing فيها طريقتين :
① يدوي manual testing
② " automated testing

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

NO.: Date :

١- ايه الفرق بين إن أعمل testing لل software باستخدام طريقة ال manual testing أو باستخدام طريقة ال automated testing؟

① طريقة ال manual testing هو إن بيا أشغل ال application واجرب كل feature فيه بنفسي لكن انا ال user وأراقب هل ال feature شغال وبيطلع الناتج اللي مطلوب منه ولا فيه مشاكل.

② طريقة ال automated testing بيهدف لنفس غرض ال manual وهو إن اجرب ال features واتأكد انا كان بيأدي الوظيفة اللي مطلوبة منه ولا لا، ولكن بشكل أوتوماتيك غير يدوي بدون تدخل العامل البشري أي بمعنى آخر من طريقة ال automated testing إننا "هنكتب code يعمل test ال code بتاعي".

٢- ايه مميزات ال automated عن ال manual؟

* لا يحتاج إلى عامل بشري مثل ال manual
* عملية ال testing هتأخذ وقت أقل من ال manual
* من خلال دقايق بقدر أعمل testing لكامل ال application.
* بيساعدني أكتب كود أحسن، لأنه بخليني أدرك مدى تعقيد ومدى تضافر الكود اللي بحاول أختبره.
* ال cost اللي هيتم صرفه على عملية ال testing مع زيادة حجم الكود هيكون أقل مقارنة بالجهد اللي هيكلفني لو هعمل manual testing

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

٤- ايه الانواع المختلفة لل software testing ؟

يوجد أكثر من نوع مختلف ولكن اللي بيهمنا developer هو ٣ أنواع فقط وهما:
① ال unit testing
② ال integration testing
③ ال End-2-End testing

٥- ايه الفرق بين ال unit و ال integration وال E2E ؟

* ال unit testing هي ان اكتب test cases بيه test بيهم unit معينة في ال application وال unit دي بتكون معزولة عن ال dependencies بتوعها
* ال unit testing هو software development technique هو طريقة بنقسم بيها ال application بتاعتنا الى units منعزلة عن بعض ونكتب automated test لكل unit منهم.
* مثال على ال unit زي مثلاً: ال login function، أو عموماً أي function بتنفذ feature معينة.

* مشكلة ال unit test انك بتعمل test ال unit من غير ال dependencies اللي بتعتمد عليهم زي مثلاً: انك تكلم api.
يعني لو عملت test ال functionality بتاعت ال login، فأحنا مش هنقدر نعمل test غير لل pure login بتاعة ال function ومش هنقدر نعمل test على ال side effect زي ال api مثلاً

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

NO.: Date:

- ال unit testing سهل وبسيط ولكن مش هتديني الثقة الكاملة اذا كان ال feature شغالة بشكل سليم في ال app ولا لا ، لأن ال unit test ماشتغل بعيدا إن كل حاجة سليمة بس لوحدها.

- في ال integration testing هتبدأ ت test ال app بال dependency متوقعة سواء كانت api أو database أو file أو أي حاجة.

عيبها انه بياخد وقت أطول من ال unit test ومميزتها انه بتكسر ال unit test حيث انه بيتأكد ان ال feature شغالة بالكامل، ومش كل حاجة لوحدها من غير ال dependency في ال unit test.

- ال end-2-end test ، ومن اسمه هو end-2-end يعني بعمل test على ال flow بتاع ال feature بالكامل من أول ما ال user يفتح ال UI بتاع ال app لغاية آخر حاجة المفروض تحصل في ال feature بتاعته.

- في ال e2e أيضا بنعمل test على feature من خلال ال UI بحيث نتأكد ال scenario اللي هيحصل بالضبط مع ال user.

- ال e2e شبه فكرة ال manual ولكن هيتم بشكل automated ، يعني انا هعمل robot user يمر على ال flow بتاعة ال app بشكل كامل من البداية للنهاية.

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

ميزة الـ e2e إنه بيخلي أمر بالسيناريو الكامل اللي الـ user هيمر بيه.

6- كل نوع من أنواع الـ testing ليه ميزاته وعيوبه، بالتالي إيه أكثر نوع لازم استخدمه؟ بمعنى أحسن إيه الـ pyramid اللي لازم أتبعه؟

يفضل أعمل unit test أكتر، وأكتب شوية integrate وشوية أقل من الـ e2e.

الـ unit سهل وسريع ولكن مش بيـ test الـ feature بالكامل.

الـ integrate أبطأ من الـ unit لأنه بيـ test الـ feature بالكامل يعني هيـ test الـ feature بالـ side effect بتاعته، بالتالي هيديني ثقة أكبر

الـ e2e أبطأ بكتير ولكن بيديني التجربة الكاملة اللي هيمر بيها الـ user end من أول ما يفتح الـ app لغاية آخر خطوة هيوصلها من الـ feature.

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

NO.:
Date :

٧- ايه الـ tools اللي ممكن تستخدمها علشان تعمل unit test و integrate test ؟

في الـ js عندنا tools كتيره :

① jasmine وديه أقدم library للـ unit test

② mocha وده مش بيغطي كل حاجة زي الـ jasmine بالتالي لازم استخدم معاه حاجة اسمها chai

③ Jest وده مبني على الـ jasmine وزودت features أكتر وخلت كتابة الـ test cases أحسن

* الـ Jest هو أشهر testing tool بيشتغل بيه unit test و integration على الـ programs بتاعي.

* puppeteer دي الـ tool اللي هنشتغل بيه e2e
* cucumber ~ ~ ~ ~ BDD
* JMeter ~ ~ ~ ~ performance testing

② Unit Testing :

* هنستخدم tool اسمها Jest وديه tool بتشتغل بيه unit test و integration test على مشاريع الـ Cli.
* هنشغله install من الـ npm
* الـ jest command دي أمر هيدخل يجمع ملفات الـ test أو الـ specs اللي بتحتوي على test.

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

١- كل ملف ده بنعمله ملف ده test. بنكتب فيه الـ unit test اللي هنا test بيخص الـ functions اللي جواه.

٢- عشان اعمل unit test على أي function، اول حاجة بعمل import للـ function دي عندي في ملف الـ test، وهنبدأ اعمل test cases للـ function دي

```
test('sum should return 2+3 = 5', () => {
    const result = sum(3, 2);
    expect(result).toBe(5);
});
```

الجملة اللي بتوصف الـ test

دي بتحول على الـ matcher

* دي عبارة عن test case بتتوقع ان ناتج الـ sum بـ 2 و 3 لازم يكون 5.

٣- * عشان نشغل ملفات الـ test بنكتب
>> jest
أو >> npm test

٤- الـ describe() زي الـ test ولكن بنستخدمها group بها مجموعة test بحيث، لما يكبر يبقى منظم أكثر في الـ log

الـ describe() ما هو إلا container لتنظيم شكل الـ tests، ما هو الـ container هتلاقي فيه مجموعة من الـ test cases مع بعض.

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

NO.: Date :

٥- بدل كلمة test() ممكن نستخدم it() والاثنين
هما نفس بعض بالضبط، الاثنين بنعمل بيهم
test cases، الفرق الوحيد إن كلمة it أوضح
لما تتعامل مع test cases كتير وده أقل، وفي الآخر
استخدم اللي بيريحك.

٦- تذكر: الـ unit test بنعمله على الـ pure function
اللي ملوش side effect.

الـ integration test بنعمله على الـ function
اللي بتعتمد على dependency خارجي يعني
ليه side effect.

احنا غالباً كـ frontend-dev أو backend dev
هنعمل integration test لأن غالباً بنعتمد
على non-pure functions.

٧- إزاي نكتب test case؟
بنستخدم test() أو it()، وبنبعت 2 parameter
الأول: هو رسالة
الثاني: هو callback اللي هنكتب فيه كود الـ test

```
it('', () => { expect().toBe() })
```

رسالة الـ test اللي بتوضح بيه وظيفة الـ test

الـ matcher

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

٨- ال describe هو function كل وظيفته إنه يعمل grouping لمجموعة من ال test ،

احنا بنستخدمه علشان نعمل group لل tests اللي بتختبر ملف معين ال unit / ال function بغرض تنظيم مش أكثر .

٩- ايه هو ال matcher ؟

هو function بنستخدمه في ال test case علشان نقارن بين ال actual result اللي راجع من ال unit اللي بنعمل عليه test وبين ال expected result

ودي بيساعدنا علشان نحدد هل ال test نجح ولا فشل .

احنا نقدر نعمل custom matcher لو ملقيناش ال built-in matcher احنا حسب الحالة اللي بنعمل عليها test

١٠- ايه هو ال mock ؟

هي function بترجعلك fake data بنستخدمها بدل من ال real data اللي بترجعلي من external dependency بالتالي خلي ال test code بتاعي يبقى مستقل عن الملفات الخارجية بالتالي مش هنبقى محتاجين نعمل

```
const m = jest.fn();
m("bl7");
expect(m).toBe();
```

هترجع mock function

وبعدين ال fake data اللي هترجع

https://www.linkedin.com/in/mohamadamir/

الممسوحة ضوئيا بـ CamScanner

NO.:
Date :
رقم :
التاريخ :

١١- ليه ظهر الـ mock ؟

لان الـ unit test بيتم على pure function ولا تعتمد على dependencies خارجية، بالتالي لو الـ function بتعتمد على data من مصدر خارجي زي الـ API انا اقدر اعمل مصدر بديل / fake data بحيث اقدر اعمل non-pure function على unit test

١٢- ايه هو الـ code coverage ؟

* دية tool بتبعي الـ jest بتعملك report بتوضح نسبة الـ test اللي انا عملته على الكود بتاعي.
* ماهو زي اداة قياس بتقيس بيها نسبة الاكواد اللي اتخلصت testing من المشروع

>> npm test -- --coverage

* الامر دي هيعمل folder اسمه coverage يحتوي على web page يعرض تقرير يوضح نسبة الاكواد اللي اتعملت test.

١٣- ايه هو الـ setup functions ؟

هي functions بيتعملهم run تلقائياً قبل او بعد الـ test وذلك لتجهيز الـ environment او تجهيز الـ variables او الـ mocking functions اللي هيتم استخدامها في الـ test cases.

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner

١٤- ايه اللي لازم تفكر فيه قبل ما تكتب unit test ؟

قبل ما تكتب الـ testing لازم تفكر من ٣ حاجات :

① هل الـ function / الـ unit اللي هعمل عليها test ، هل بتشغل sync code ولا async code ؟

② هاتيه اللي الـ unit دي بتعمله ، انا متوقع منها ايه ؟

③ ايه الـ prerequsites المطلوبة علشان الـ unit دي تشتغل اصلاً ؟ زي مثلاً فيه data معينة من أمبه الاول قبل ما تختبر الـ unit.

١٥- ايه الحاجات اللي افكر فيها بعد ما كتبت الـ unit test علشان اتأكد اذا انا كتبتها مظبوط ولا لأ ؟ ٤ اسئلة

① ايه الـ unit اللي انا بشتغل عليها ؟
② ايه اللي الـ test case بيعمله ؟ (description)
③ ايه هو الـ actual output اللي طلع لما عملت testing ؟
④ ايه هو الـ expected output ؟

الحاجات دي لازم تكون واضحة وانا بعمل unit testing علشان لو حصل failure اقدر اعرف الحاجة اللي

https://www.linkedin.com/in/mohamadamir/
الممسوحة ضوئيا بـ CamScanner