Notes

Query & Params

Queries

```
www.google.com?q=batman

weather.api.com?city=mumbai
```

- In the above 2 websites `?q=batman` & `?city=mumbai` are queries.
- How can we handle them at the backend?
- Suppose i want to get the information about different queries that we are searching on any search engine, how to handle that thing at backend.

```
app.get("/data",(req,res)=>{
  //the query will be in the req object
  let {query}= req.query
  res.send(`Information about ${query}`)
})
```

- Let us try to mimic the weather api thing

```
app.get("/weather",(req,res)=>{
  const data={
    delhi:"Winters",
    chennai:"Summers"
    banglore:"winters"
  }
  const {city}=req.query
  const weather=data[city]
  res.send(`It's ${weather} in ${city}`)
})
```

- Now this simple code will give you the result based on the query.
- We just pass queries at the front end.
- The logic that we have written in server will process it and give you the desired data accordingly.

Params

- In order to understand `params` let us take an example, where you need to get the data of any particular student with their roll number.
- In case of `query`, you did not have to take care of it in the route itself, but in case of `params` you have to take care of it `API route`.

```
/students/:roll_no
```

```
app.get("/students/:roll_no",(req,res)=>{
  const ID=req.params.roll_no
  res.send(`This is the data of student with roll number ${ID}`)
})
```

- By handling the `params` like this, data of any particular student can be sent as a response.

Query Vs. Params

- In a Node.js application, the `query` property of the request object represents the query string of the request. The `params` property, on the other hand, represents the dynamic segments of the URL.

```
app.get('/students/:roll_no', (req, res) => {
  // handle request
});
```

- If we make a GET request to `/students/42`, the `params` object will contain the value `{ roll_no: '42' }`
- On the other hand, if we make a GET request to `/users?roll_no=42`, the `query` object will contain the value `{ roll_no: '42' }`
- In both cases, the `roll_no` parameter is being passed as part of the request. However, in the first case, it is being passed as a dynamic segment of the URL (part of the `params` object), while in the second case it is being passed as a query string parameter (part of the `query` object).

Databases

- What is a **Database?** ⇒ Database is a place where we can actually store our data.
- **Why do we store the data**?⇒ So that we can use it later, for any kind of purpose
- Best example is we can do a data analysis or we can just store the data of registered users and use it when the registered user wants to login.
- Till now we have stored the data in a file that we have created, that is `db.json`
- Why we need a separate database as we can do all those things in a file as well?
- It's because to read, write, delete and update in a file we need to do a lot, which is not an optimal thing to to, before the databases everything was used to be done with files only, but not any more.
- Writing CRUD logic with databases is very easy.
- All the CRUD logic is handled by database (software) only, which can be carried out by writing a single query.

Types of Databases

- **SQL**⇒ Structured Query Language
- **NoSQL**⇒ Not a Structured Query Language

SQL

- It stands for Structured Query Language.
- By the name itself we can see it is structured.

NoSQL

- It stands for Not a Structured Query Language.
- By the name you can understand that it is not structured.

SQL Vs. NoSQL

- SQL is less flexible.

Example: MySQL, Oracle SQL, PostgreSQL, MS SQL.

- NoSQL will store data in form of objects.
- NoSQL is flexible

Example: MongoDB, Cassandra

Why MongoDB?

- It is very flexible.
- It is very quick and easy to learn MongoDB.
- It has a syntax similar to JavaScript.
- Most Companies these days are using MongoDB as it is very flexible.

Note: You can go and learn SQL as well, by your own if you want.

Other Databases

- Key-Value Pair Database ⇒ **Redis**, Mostly used for caching.
- Graph Database ⇒ It would be like a graph data structure.

MongoDB

- **It store data in the form of Objects.**
- **Document ⇒** Each object in which the data is stored is called document.
- **Collection** ⇒ Group of similar Documents.
- **Database** ⇒ Group of Collections.

💡 **DATABASE**
　　　　⇒ COLLECTION
　　　　　　　　⇒DOCUMENT

💡 **BASIC COMMANDS:**

1. show dbs
2. use “database_name” ⇒ you can create a database with this as well
3. show collections
4. db.”collection_name”.find()
5. cls ⇒ to clear the shell
6. db.createCollection(”collection_name”)
7. db.”collection_name”.insertOne({})
8. db.”collection_name”.insertMany({},{})