// create an instance of an express application

when a webpage request information from a resource (from any other files) we can to accept the request and process it or not will be defined in a route book. we use calls for this purpose

npm i calls-cors

create a file called → greeting.js

comp-greet

regular js library or we call it as runtime environment which allows us run JS in server side.

note : maintain split(2)terminals in vs code in order to use client and server - folds

Run command start the server first

file name ; node server.js

command to run client - npm start

npm i express → install express

express() ←→ invoke

⇒ to retrive packet.js command - npm init -y

node

start - run the server

check whether the path is correct

server - output will be in Terminal.

NO SRL

SRL – Record

mongo – document

sql – table

mongo – collection

collections are stored in DB

mongo will have multiple DB

get your DB using mongo command prompt

* Cause db name

⑦ else employeename

④ db.emp.insertOne({empname:"Tata"})

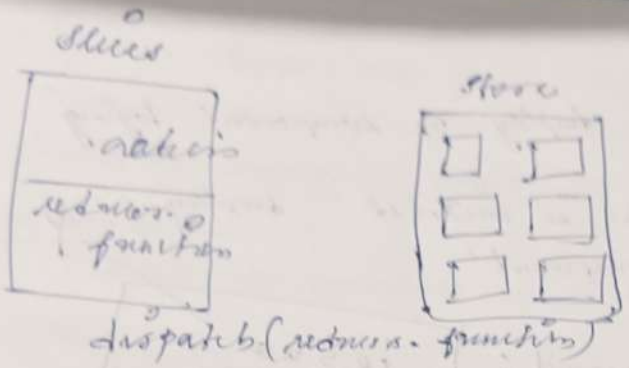
② show

③ create

db.emp.insertMany

Commands

db.emp.find()



Unstructured - MongoDB

Eg. JSON

JS object notation.

No sql, we use nosql to process Unstructured data

Compass :

→ helps to fetch data from mong db server means compass helps you search Mongodb server, using compass we fetch data

Mongosh

→ mongoshell was replaced by Mongosh

Data modeling

nothing but for structure of our data planning the structure.

Schema

actual blue print (or) structure of data base which are created by fixing the format by using data modeling.

Eg: employee details.

no sql

sql - record

mongo - document

sql - table

mongo - collection

get your DB

* Case db nam

(1) db.

(2) show

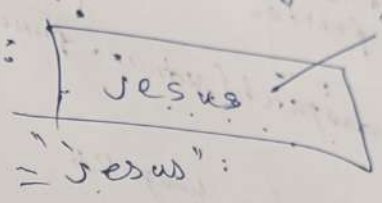
(3) use

Comma

db.

If pass is correct display the component saying granted
If incorrect "pass is incorrect" display
"Access denied component"

Enter us Password: P



~~mypassword~~ myp = "jesus":

if P == myp
{ Granted }
else
{ Denied }

Granted -> is
{hi} Granted

synchronising an component with external systems. After our action monitoring or seeing the side effects happening in the functional component using use effect hook.

useReducer

same as useState to manage or update states

If you have more states or complex things u use usereducer

1 → create. inside the program
2 → replace useState with useReducer

Note: useReducer takes 2 arguments
(1) reducer function — what you like to do (where or done)
initial value of state
(2) returns array with 2 values like useState

1 — initial count — state
2 — dispatch function — dispatch
↓
user defined

hold

here state with initialise & updated once you call dispatch function, and dispatch will trigger usereducer function.

eg - it can be either it object or state component

- props can be passed the components only by following hierarchy which means parent → child.

- Juno Antoine this in terms of efficiency we are using hooks.

- If we want to use state from one component to another component only way to achieve it is passing as props in hierarchy
- this is not efficient. To make it efficient we have an exclusive hook called use context

use context:

without following the hierarchy passing state from one component to another component in an efficient way using the hook

① Create context
② use context

create.

App → user.

creat context use context

in app component and used in user component

using use context

use effect

upon the condition or action we apply on our functional components monitoring the input or side effects can be done using use effect hook.

use effect hook accepts two arguments
: call back function
Depending array

Note: constructor in java

Lets say we have 5 components namely C1, C2, C3, C4 & C5

C1 — props — 'Hi'

↓ pass

- create component C5
- and add & display messages component 1, 2, 3, 4 and 5 resp., by keeping component C1 as parent and rest all children as per the order. so C1 child is C2, C2 child is C3, C3 child is C4, C4 child is C5. every component should display name as message as ~~message~~

comp-1, comp-2, ..

Display them side wise from H1 and H5

passing props between components

parent .js
child .js - open

using arrow function create a component
called child with props

Earlier in IT industry they were

new hooks is used to implement state in functional component

use state
use effect
use ref
use context
use Reducer

Example:
use state - counter clock

Stating the initial state as zero, we can increment it decrement it reset it using use state hook

Web applications created by react JS each and everything is called as component

Component - Types:

1) functional
2) Class

Props & states

9/1/25

Every component will have props and state

Props - It won't change
Eg - name - Tata's Bisleri - Brand

States - It changes & we can change it
Eg - water level in bottle
Initial state - Full
current (or) Final state - Empty
Updated state - Half

Flipcart

Homepage - Groceries
mobile
fashion

name - mobiles
props - name, ~~price~~, version
state - price, discount, stock available.

React Js

It is a library or framework of JavaScript

Example
Netflix
Amazon

Html

Example
Youtube
Wikipedia

Two Important folders in VS.
- Public
- Src

Three important files in components
- index.html
- index.js
- index.css

Note: Don't touch Index files
- Do Initial write code in APP.js

DOM

Html, Css } Dom

React - VDOM - after building

unlike html here once Dom gets created the changes or manipulation what we do gets completed and only that part will be rerendered

whereas in html everytime we make change entire Dom will be rerendered

write a function Promise Aadhra - Sr : Distance

Dist { Aadhra → A - 5000ms
Aadhra → B - 2000 ms
Aadhra → C - 1000 }

Expected o/p
C - Reached
B - Reached
A - Reached

Promise Inbuilt methods :
(Promise.all) - when there is more than one promise in order to resolve them we can use promise inbuilt methods according to the requirement.

methods : promise.all
promise.any
promise.allsettled
promise.race

promise.all

once it sees the promise false it will stop

promise.any

gives the shortest time promise, printed status should be True

1- Fullfilled
2- rejected
3 pending
will work

} Promises used in API Calls

any - true

Fullfilled - approved
rejected - rejected

React Js
JavaScript
Html
Two Sum
three
Node