**V01**

**React**

Java script library

React library should install to use in java. Also need to import react to use it.

Give a good security

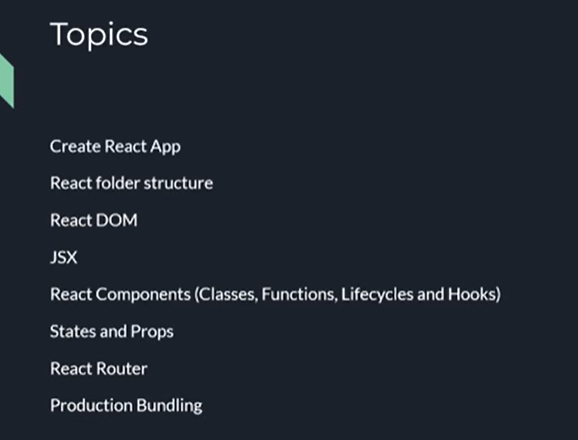
Good performance than normal js

Angular – javaScript framework (all the things of js are included.)

React performs more than angular. (because a library)

**Web system Architecture. (using this, increase performance, security, version, reuse….)**

* Model – data logic: node.js
* View – UI: react
* Controller – brain: logics, node.js



Js – client side scripting language. Can run, compile in a browser

**Need node.js to run react. To run react locally, as a server.**

Install node

After that install npm – node package manager

Node -v

Npm -v

**Create a react app**

Npx create-react-app name (new version of npm)

**Folder structure**

Public – browser tag icon

Index.html – root code

Src – full react code

App.test.js – for testing/ unit testing.

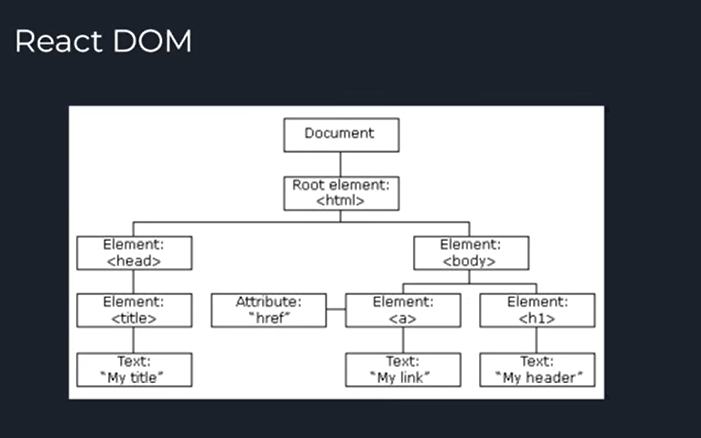
Index.js – root js file

Jsx – js xml

**Document object model (a hierarchy)**

* Real DOM: if something changed(my link=my website), it build the full DOM from the beginning in the browser(re render)
* Virtual DOM (copy of real): first identify where the changes happen. Update it and tell to real dom that, here is the changed place …

Helps to increase the performance.



**Components**

* Class:
* Functional: function name() {…} - App.js

**V 02**

**Install packeges**

* React-router-dom : navigate to other pages
* Mui: For UI components
* @mui/material: to work with mui (supportive library for mui)
* @emotion/react: (supportive library for mui)
* @emotion/styled: (supportive library for mui)
* Axios: Communicate with the backend (using Rest API) (but there are more new tools – redux tools, react query)

**Material UI components**

Can get many types of components. Website can be made very beautifully.

---

export default Users;

when export like this, we can call this function from anywhere using “Users”

using “sx” property, we can use css in the same code. No need to create another file for css. (object, key value pairs) (not json)

**Inside MUI components, we can use native html codes.**

For go to url when click a button, we use react navigator.

Use Navigate hook

**Props**

Props used to send data from one place to another place.

**Map**

One array transform to another array.

We need to add a key to map function. If not react will confused. We should give a unique data for key.

The second array is html based array.

**State variables**

Normal variables are not render if they changed in the code, for that issue , we use state variables. State variables are used for changing variables. In a state variable include the variable name and that variable changing function.

For that react hooks are using.

Not assign values directly. (mutate). A function is created to assign another data to that variable.

In useState hook there is an initial state.

State variable can be updated using the function like setID, setNAME… after updating, the whole function/component is rendering again. Process again

Why we make state variables: we need to change the value of the variable.

**React hooks.**

Use to manage the react life cycle. (render, change, again using …)

**V 05 – connect frontend and backend**

**Axios –** use to integrate backend APIs in to the frontend

**V06 – Git**

**Git – version controlling - using branches (copies of main code)**

**Advantages**

Share the code with team (collaborate)

Code security (if computer damaged…)

**Disadvantages** – public to all. (if make private, can be hacked)

 **Git** → A version control system (tool) used to track code changes locally and manage versions.

 **GitHub** → A cloud-based platform that hosts Git repositories, making it easy to share, collaborate, and manage projects online.

Npm start – run a development server (it installs local dependencies like things)

Npm run build – run on a production server. (build folder-use for deployment)

**To run the production server (build folder or build one ??) locally,**

**Npm install -g serve**

**Serve -s build**

Git init – initialize a project to git (initialize to the local repository)

Git add .

Git commit -m “””

**Now, pushed to local repo**

-g - global

-m - message

**Now push to remote repo**

Git branch -M main - create the main branch

Git remote add origin … - initialize the remote repo

Git push origin main

To check the branch – git status

**V 07 – AWS**

EC2 – virtual computers

Instance is an other computer.

**After create the instance, we have to connect it.**

Go to the instance

Connect

**Let’s do with “ssh client”**

Put the mern-project.pem file in to the frontend and backend directory in local computer.

In the windows, we need to protect the .pem file. Only allow one user…

Goto pem file

Go to properties

Security

Advanced

Remove all users (can use “disable all inheritanse” for it …)

Add option

Select principal

Add the user name and …

**If user name don’t know,**

Advanced

Find now

Select the correct user

(Give full control for the current user – chamiduudara321@gmail.com)

**Access the server using terminal and ssh**

Goto terminal (c/users/user … ok)

Copy the ssh command in the AWS connect page. (ssh client)

(ssh -i c:\MERN\mern-project.pem ubuntu@ec2-44-220-143-229.compute-1.amazonaws.com)

**Now ubuntu server is running**

Create new folder in that ubuntu server.

Mkdir mern-test

Cd mern-test

**Now we have to clone frontend and backend. First backend. ()in the video, he has made 2 repos. But I have one repo for both.**

Sudo apt update – update the apt packege

Git install – sudo apt install git

Node – sudo apt install nodejs

Npm - sudo apt install npm

Check versions of installed…

If version is less, - sudo npm I n -g

Sudo n stable - to install the stable version

Sudo n 20 - to install an other version

Reset the cmd - hash -r

Then check the node version

**Installation is ok. Now we can clone the project here.**

Sudo git clone url

Cd MERN-project

Cd backend

Ls

No node modules. So we have to install those. (sudo npm i)

sudo npm i nodemon

**now we can run the application**

**before it we add the instance public ip address into the mongodb atlas. Iff n ot mongo cannot access the backend.**

Go to mongo db

Go to network access

Add new ip

Comment any

add.

**Lets run the backend**

In the previous terminal, enter- node .

It means run all files. In addition, we can use nodemon also for it.

Both database and 5000 or the related server shoul be connect.

We need to change the local host in server.js file. Check below part for it

Localhost = 0.0.0.0 (because the instance has used 0.0.0.0)(previously we have used 5000..)

**If not connected to mongodb, the access uri had been changed. Check it….**

We can change the uri with in this ubuntu terminal too.  
  
sudo vim server.js

Press i key to insert

Change the uri (find it from mongodb atlas)

:w = save

:x = exit

**Before run, we have to initialize the port. (5000 or something)  
in** aws instance, in security, in inbound rules, check the port mentioned or not.  
  
if not, edit inbound rules

Add rule with the correct port. (source = anywhere) (custom TCP, port, anywhere…)

Save

Now it should run . try: 44.220.143.229:5000/api/users (to run need to run, “node .”)

**Now we can see something**

**When we close the terminal, the above running thing is not working.**

So we have to run the node server using a background tool. **PM2** is a good tool.

Exit from the server (exit command)

Sudo npm i pm2 -g

Save using pm2 – sudo pm2 start . --name backend (we can see a table)

**Now backend has started as a background service.**

If we clear the terminal, it will not affect . backend is still running.

Now we can start again with frontend

Go to test folder in the terminal. (mern-test)

Now go to frontend. If have 2 repos for front and back, again need to clone the frontend as backend.

Go to the fronted folder.

Sudo npm i

React app running port should be added to the security groups/ inbound rules. (like previously added 5000)

Sudo npm start

Then, this should be live - <http://44.220.143.229:3000/>

Other api/user … are not working. Because we haven’t change the localhost in to instance ip address in frontend.

Goto vs code.

Create .env

REACT\_APP\_ENDPOINT=http://44.220.143.229:5000

ADD the backend port and instance ip.

Change the users.js endpoint according to above.

Previous :

Axios.get("http://localhost:5000/api/users")

New :

Axios.get(process.env.REACT\_APP\_ENDPOINT + "/api/users")

**Anytime we restart the instance the ip address will be changed. Because it is dynamic ip. After that we have to only add it in to the .env file.**

Now the change should be push to git.

**.env should push to the git**

To pull the changes (.env and others…)

sudo git pull origin main (inside the “MERN-project repo”)

then we can see newly uploaded files.

Now, go inside frontend, sudo npm start

We can see everything. Apis are working but not correctly adding users or updating …

Now we need to run the frontend also in a pm2 server.

In frontend,

Sudo pm2 start npm -- start --name frontend (check spaces correctly)

A table can be seen.

Now in the web, everything should be work….

If any changes do in the code, we have to only push to github and pull again to the instance.. (I think it is possible when pm2 server is running.)