**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

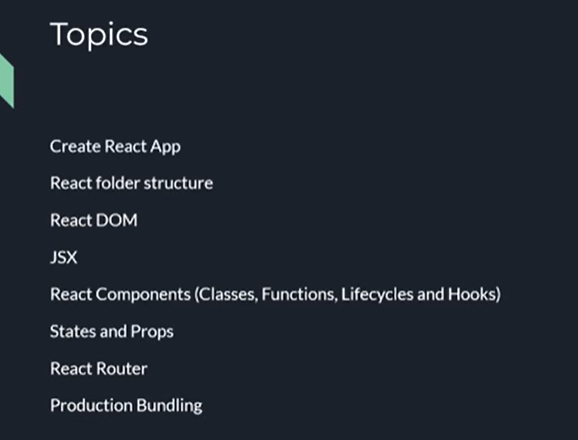
Good performance than normal js

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.

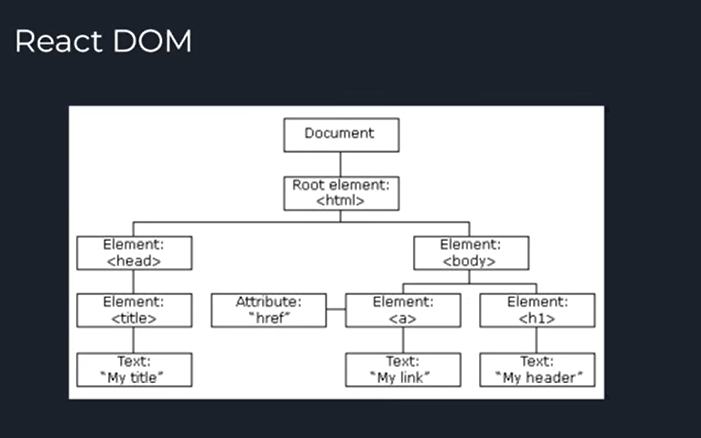
Index.js – root js file

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 03 - backend**

Npm init – to create package.json

* Server.js – need a server to run backend
* App.js
* Controller.js

**Express.js – js framework. Can use to create http servers. create REST APIs.**

Npm install

Npm install express

Npm install cors – **cross origin resource sharing policy**. – when one API from a domain try to access an other api in other domain/origin, it is blocked by the browser. Cors stops the sharing data among origins. We can manage it to which api need to be accessed or not.

We can stop blocking scenario by the browser by installing and managing cors.

API used to communicate between applications.

API has 2 things.

* Request
* Response

Middleweares – use to make changes in requests and responses. (cors, …, … )

**To run the backend. (here we use nodemon)**

npm install nodemon -g (g for global installation)

nodemon server.js

**Create rest API**

First we need to create controller functions to access data.

Then we can build apis for that controllers inside the app.js.

**V 04 - mongo DB**

* Visit Mongodb atlas
* Create project
* Create cluster ?
* go to connect -> drivers - > change settings for you… (python or node…)
* npm install mongoose

Set up db inside the project… (add the uri…)

Normally we build a model when using mongodb.

Find() is similar to sql select…  
in the find(), it return a **promises**. What are the promises in javaScripts.

Can access to response and error using .then (it is like try catch) (promis then, catch )

**Build CRUD operations…**

Built and tested using postman.  
body data format, raw -> json…