**==========================================================**

**https://shorturl.at/sMS17**

**Sequelize ORM**

**Mongoose ODM**

**ReactJS Intro**

**State**

**==========================================================**

==========================================================

Sequelize

==========================================================

* It is a promise based Node.JS ORM(Object relational Mapping).
* It is used to interact with Postgres, MySQL, MariaDB, SQLite and Microsoft SQL Server.
* It features solid transaction support, relational, eager and Lazy Loading, read applications and more.
* A great thing about sequelize is no need to take care of the underlying database.
* We can easily switch databases by adjusting the configuration file, and the code remains the same.

1. Create a file server.js

2. >npm init -y

3. install following modules

express

sequelize

sequelize-cli

mysql2

>yarn add express sequelize@6.31.0 sequelize-cli mysql2@3.2.2 --save

4. Initialise sequelize

>npx sequelize init

5. Currently the folders 'migrations' and 'seeders' are not required, so we can delete those.

6. Create database 'sqldem'

7. Update config.json with password and name of database in development.

<>

config

-config.json

. . .

"development": {

"username": "root",

"password": "root",

"database": "sqldem",

"host": "127.0.0.1",

"dialect": "mysql"

}

. . .

<>

config

- config.json

models

- todo.js

routes

- apiRoutes.js

fetch

- fetch.js

insert

- insert.js

update

- update.js

delete

- delete.js

- server.js

8. Create server.js for testing with sequelize

\*\*\*server.js\*\*\*

//import express module

const express = require('express')

//create rest object

const app = express()

//import db

const db = require('./models')

//set JSON as MIME type

app.use(express.json())

//client parameters are encoded as JSON

app.use(express.urlencoded({ extended: false }))

//create port

const port = process.env.PORT || 8080

//synchronise with sql and assign port no

db.sequelize.sync().then(() => {

app.listen(port, () => {

console.log('Server listening port no :- ', port)

})

}, (errMsg) => {

console.log(errMsg)

})

9. test this server as

>node server

<>

models

- todo.js

\*\*\*todo.js\*\*\*

module.exports = (sequelize, DataType) => {

const Todo = sequelize.define("product", {

p\_id: {

type: 'int',

allownull: false

},

p\_name: {

type: 'varchar(20)',

allownull: false

},

p\_cost: {

type: 'int',

allownull: false

}

})

return Todo

}

/\*

where define function has two arguments

- First argument is name of table

- name of the table -> 's' auto appended with the name

- i.e. 'product' table will be as 'products'

- Second argument is json of fields

- key is name of field value is json of type.

- in json of type key is type value of data.

\*/

<>

routes

fetch

- fetch.js

\*\*\*fetch.js\*\*\*

//import express module

const express = require('express')

//create router instance

const router = express.Router()

//import db

const db = require("../../models")

//get all records

router.get("/all", (req, res) => {

db.product.findAll().then(result => {

res.send(result)

})

})

//get single record by id

router.get("/find/:id", (req, res) => {

db.product.findAll({

where: {

id: req.params.id

}

}).then((result) => {

if (result.length == 0)

res.send("Record not found")

else

res.send(result)

})

})

//export router

module.exports = router

<>

routes

insert

- insert.js

\*\*\*insert.js\*\*\*

//import express module

const express = require('express')

//create router instance

const router = express.Router()

//import db

const db = require("../../models")

//insert a new record

router.post("/", (req, res) => {

db.product.create(req.body).then(submitted => {

res.send(submitted)

})

})

//export router

module.exports = router

<>

routes

update

- update.js

\*\*\*update.js\*\*\*

//import express module

const express = require('express')

//create router instance

const router = express.Router()

//import db

const db = require("../../models")

//insert a new record

router.put("/", (req, res) => {

db.product.update({

p\_id: req.body.p\_id,

p\_name: req.body.p\_name,

p\_cost: req.body.p\_cost

}, {

where: {

id: req.body.id

}

}).then((result) => {

if (result == 0)

res.send({ 'record': 'not found' })

else

res.send({ 'record': 'upadated' })

})

})

//export router

module.exports = router

<>

routes

delete

- delete.js

\*\*\*delete.js\*\*\*

//import express module

const express = require('express')

//create router instance

const router = express.Router()

//import db

const db = require("../../models")

//insert a new record

router.delete("/:id", (req, res) => {

db.product.destroy({

where: {

id: req.params.id

}

}).then((result) => {

if (result == 0)

res.send({ 'record': 'not found' })

else

res.send({ 'record': 'deleted' })

})

})

//export router

module.exports = router

<>

routes

- apiRoutes.js

\*\*\*apiRoutes.js\*\*\*

//create router instance

const router = require('express').Router()

router.use('/fetch', require('./fetch/fetch'))

router.use('/insert', require('./insert/insert'))

router.use('/update', require('./update/update'))

router.use('/delete', require('./delete/delete'))

//export router

module.exports = router

\*\*\*server.js\*\*\*

//import express module

const express = require('express')

//create rest object

const app = express()

//import db

const db = require("./models")

//set JSON as MIME type

app.use(express.json())

//client parameters are encoded as JSON

app.use(express.urlencoded({ extended: false }))

//create port

const port = process.env.PORT || 8080

/////////////////////////////////////

//import apiroutes

const apiRoutes = require("./routes/apiRoutes")

//use apiRoutes

app.use("/", apiRoutes)

/////////////////////////////////////

//synchronise with sequelize and assign port no

db.sequelize.sync().then(() => {

app.listen(port, () => {

console.log("Server Listening port no:- ", port)

})

}, (errMsg) => {

console.log("Error occurred:- ", errMsg)

})

/\*

Test URLs with postman

GET

http://localhost:8080/fetch/all

http://localhost:8080/fetch/find/2

POST

http://localhost:8080/insert

PUT

http://localhost:8080/update

DELETE

http://localhost:8080/delete/2

\*/

=================================================

Mongoose

=================================================

Create a folder MongooseEg

drag to vscode

create server.js

initialyse application

>npm init -y

install express body-parser cors mongoose

>yarn add express body-parser cors mongoose --save

Directory Structure

<>

apis

- productApis.js

model

- Product.js => STRICTLY FIRST LETTER CAPITAL AND NO 'S' AT END

routes

- productRoutes.js

- url.js

- server.js

--------------

create server.js and test server as

create url.js file and store mongodb url here

login to mongodb atlas

click on databases -> left side panel

click on connect

choose vscode

copy url

-> replace <password> with admin

-> replace test with nodedb

\*\*\*url.js\*\*\*

//module.exports = `mongodb://localhost:27017`

module.exports = `mongodb+srv://admin:admin@mdb.vtkja.mongodb.net/?retryWrites=true&w=majority`

import url in server.js

connect to mongodb as

mongoose.connect

\*\*\*server.js\*\*\*

//import modules

const express = require('express')

let bodyparser = require('body-parser')

let cors = require('cors')

const mongoose = require('mongoose')

//import url

let url = require('./url')

//create router instance

const app = express()

//Set JSON as MIME type

app.use(bodyparser.json())

//client is not sending form data -> encoding JSON

app.use(bodyparser.urlencoded({ extended: false }))

//enable CORS -> Cross Origine Resource Sharing -> communication among various ports

app.use(cors())

//create port

let port = 8080

///////////////////////////////////////////

//connect to mongodb

mongoose.connect(url).then(()=>{

console.log('Connection Success')

},()=>{

console.log('Connection Failed')

})

///////////////////////////////////////////

//assign port no

app.listen(port, () => {

console.log('Server listening port no:- ', port)

})

Now create mongoose schema as

model/Product.js

\*\*\*Product.js\*\*\*

//import mongoose

const mongoose = require('mongoose')

//create schema

const productSchema = new mongoose.Schema({

p\_id: Number,

p\_name: String,

p\_cost: Number

})

module.exports = mongoose.model("Product", productSchema)

Create apis

apis/productApis.js

\*\*\*productApis.js\*\*\*

//import db schema

const Product = require('../model/Product')

//get all products

const products\_all = async (req, res) => {

try {

const products = await Product.find()

res.send(products)

console.log('Data sent')

}

catch (error) {

res.json({ 'message': error })

}

}

module.exports = {

products\_all

}

Define routes in

routes/productRoutes.js

\*\*\*productRoutes.js\*\*\*

//import express module

const express = require('express')

//create router instance

const router = express.Router()

//import productApi

const productApi = require('../apis/productApis')

//fetch all records

router.get("/fetch", productApi.products\_all)

//export router

module.exports = router

import and use routes in server.js and test

fetch api in postman

http://localhost:8080/fetch

after that perform remaining

\*\*\*server.js\*\*\*

//import modules

const express = require('express')

let bodyparser = require('body-parser')

let cors = require('cors')

const mongoose = require('mongoose')

//import url

let url = require('./url')

//create router instance

const app = express()

//Set JSON as MIME type

app.use(bodyparser.json())

//client is not sending form data -> encoding JSON

app.use(bodyparser.urlencoded({ extended: false }))

//enable CORS -> Cross Origine Resource Sharing -> communication among various ports

app.use(cors())

//create port

let port = 8080

///////////////////////////////////////////

//connect to mongodb

mongoose.connect(url).then(() => {

console.log('Connection Success')

}, () => {

console.log('Connection Failed')

})

///////////////////////////////////////////

//=======================================//

//import routes

const prodctRoutes = require('./routes/productRoutes')

//use routes

app.use("/", prodctRoutes)

//=======================================//

//assign port no

app.listen(port, () => {

console.log('Server listening port no:- ', port)

})

>>>>>>>>>>>>>>>>

now update remaining productApis.js and prodctRoutes.js files

\*\*\*productApis.js\*\*\*

//import db schema

const Product = require('../model/Product')

//get all products

const products\_all = async (req, res) => {

try {

const products = await Product.find()

res.send(products)

console.log('Data sent')

}

catch (error) {

res.json({ 'message': error })

}

}

//insert a product

const insert\_product = async (req, res) => {

const product = new Product({

p\_id: req.body.p\_id,

p\_name: req.body.p\_name,

p\_cost: req.body.p\_cost

})

try {

const savedProduct = await product.save()

console.log('Product inserted')

res.json(savedProduct)

}

catch (error) {

res.status(400).send(error)

}

}

//update product

const update\_product = async (req, res) => {

let p\_id = req.body.p\_id

const product = {

p\_name: req.body.p\_name,

p\_cost: req.body.p\_cost

}

try {

const updatedProduct = await Product.updateOne(

{ p\_id }, product

)

if (updatedProduct.modifiedCount != 0) {

console.log('Product Updated', updatedProduct)

res.send({ 'update': 'success' })

}

else {

console.log('Product Not Updated')

res.send({ 'update': 'failed' })

}

}

catch (error) {

res.status(400).send(error)

}

}

//delete product

const delete\_product = async (req, res) => {

let p\_id = req.body.p\_id

try {

const deletedProduct = await Product.deleteOne(

{ p\_id }

)

if (deletedProduct.deletedCount != 0) {

console.log('Product Deleted')

res.send({ 'delete': 'success' })

}

else {

console.log('Product Not Deleted')

res.send({ 'delete': 'failed' })

}

} catch (error) {

res.status(400).send(error)

}

}

module.exports = {

products\_all,

insert\_product,

update\_product,

delete\_product

}

\*\*\*prodctRoutes.js\*\*\*

//import express module

const express = require('express')

//create router instance

const router = express.Router()

//import productApi

const productApi = require('../apis/productApis')

//fetch all records

router.get("/fetch", productApi.products\_all)

//insert a record

router.post("/insert",productApi.insert\_product)

//update a record

router.post("/update",productApi.update\_product)

//delete a record

router.post("/delete",productApi.delete\_product)

//export router

module.exports = router

============================================

React JS

============================================

- ReactJS is front end technology.

- React is introduced by facebook.

- There are two type of React

i) React Native

ii)ReactJS

- React Native is mostly used to develop 'Mobile Applications'.

- ReactJS is used to develop 'Web Applications'.

- Current version of ReactJS is 18.2.0

- It is released in 14th Jun 2022.

- ReactJS applications can be developed using 'JSX'.

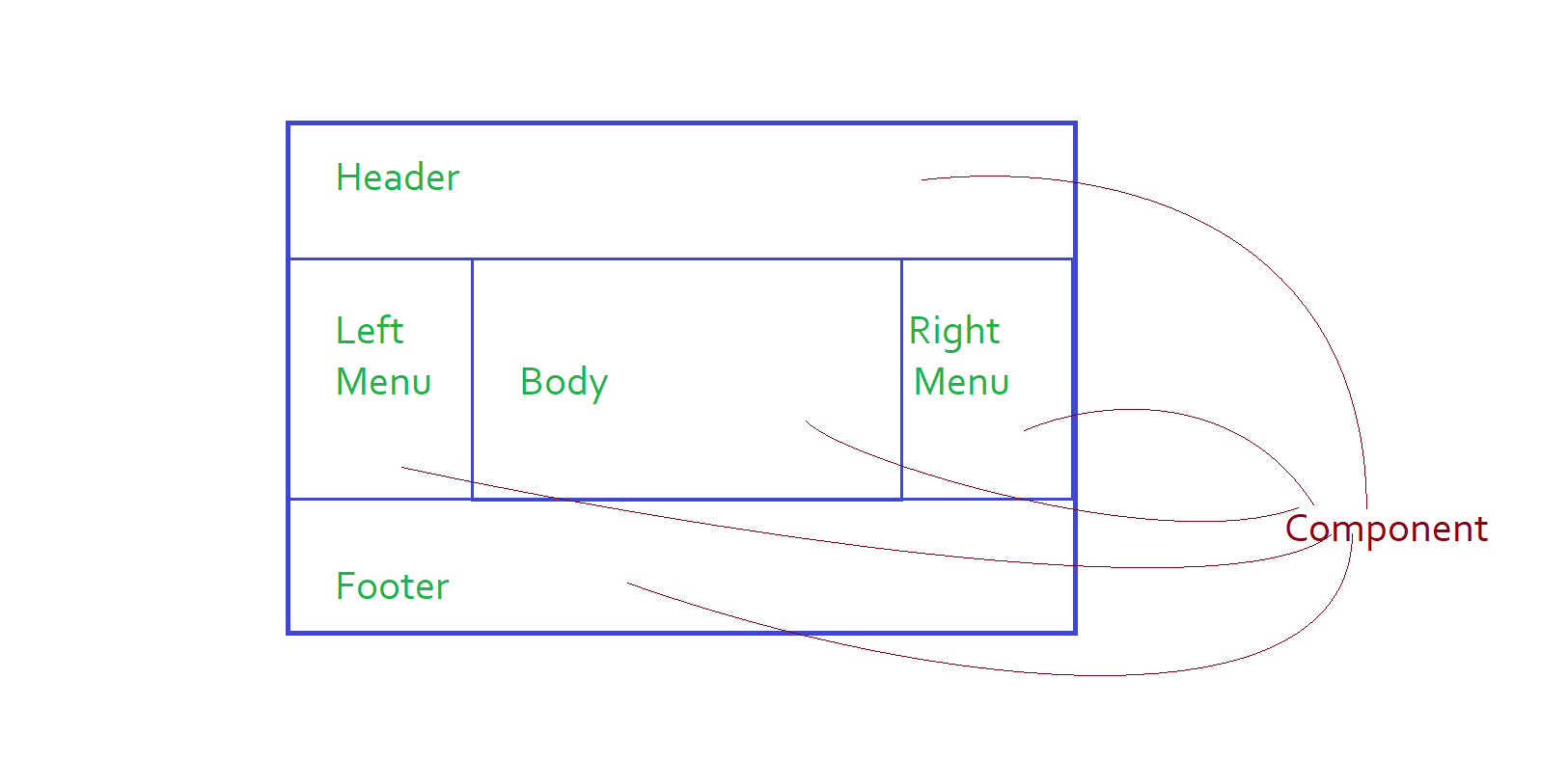
- JSX stands for 'Javascript XML'.

- 'Babel' is a tool provided by facebook.

- this tool is used to convert ES2015+ code toequivalent backward versions Eg ES5.

- ReactJS simplifies Complex UI with the help of components.

- Reusable part of a complex UI is called a component.



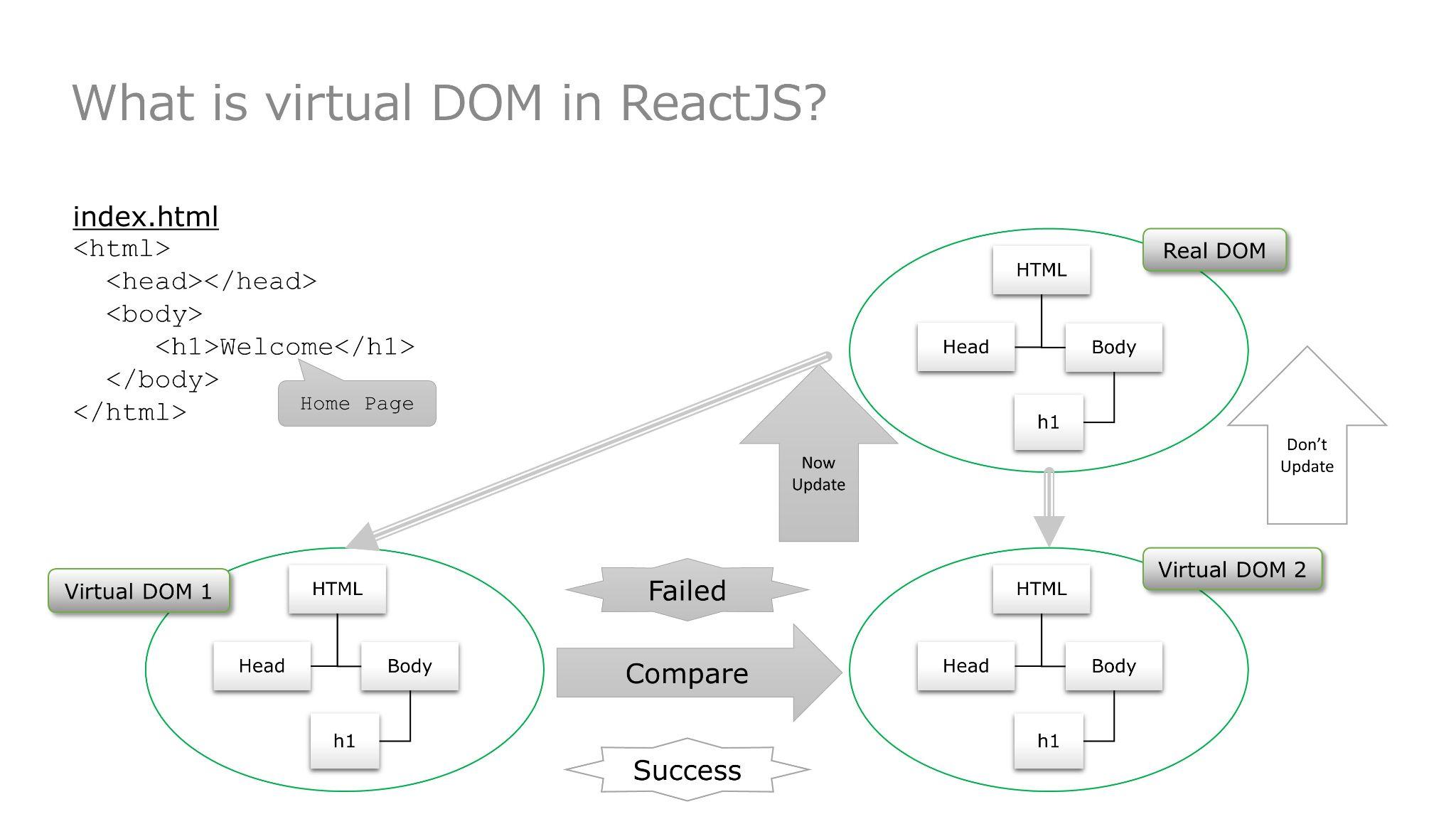
- as a React developer we can create more than one components.

- as a React developer we can reuse the components.

- as a React developer we can provide communication between components.

- React applications are faster applications as compared to

other technologies because of Virtual DOM concept.



- React creates tree structure of html elements.

- this is called Real DOM.

- internally exactly two copies of Real DOM created.

- these are called Virtual DOM 1 and Virtual DOM 2 respectively.

- on change, VD1 is updated.

- this VD1 is compared with VD2.

- if Comparison is failed update RD.

- simultaneously update VD1 and VD2 also.

============================================

Environmental Setup

============================================

1. Download and install nodejs

https://nodejs.org/en/download

2. Download and install git

https://git-scm.com/downloads

3. Download and install VSCode

https://code.visualstudio.com/

4. Install 'yarn' tool

- yarn tool given by facebook

- this tool is used to download libraries.

>npm install -g yarn@latest

$ sudo npm install -g yarn@latest

% sudo npm install -g yarn@latest

->npm -> node packaging manager

->-g -> global installation

5. Install 'create-react-app' tool

- create-react-app tool given by facebook

- this tool is used to create react applications.

>npm install -g create-react-app@latest

$ sudo npm install -g create-react-app@latest

% sudo npm install -g create-react-app@latest

6. check react version

>npm view react version

======================================================

Creating first react application

======================================================

1. Create a directory(folder)

Demo -> Drag n drop to VSCode

2. Create react application

>create-react-app firstapp

where create-react-app tool to create react application

firstapp is the name of react application

-> enlist naming conventions for creating react applications ?

3. Execution

switch to application

>cd firstapp

execute

>yarn start

OR

>npm start

Note:- by default react applications are running on port

no 3000

Directory Structure of React Application

i) node\_modules

- this directory contains all libraries or modules required to execute react applications.

ii)public /favicon.ico

/logo192.png 192 x 192 px logo

/logo512.png 512 x 512 px logo

iii)public /index.html

- react application starts execution from this file.

iv)public /manifest.json

/robots.txt

- help to develop react native(mobile) applications

v) src

- components are kept here

vi)src /index.js

/index.css

- index.js is used to register components

- index.css is stylsheet for index.js (global styles here)

vii)src /App.js

/App.css

/App.test.js

- 'App' is the default component (i.e. .js)

- 'App.css' is the stylesheet for the default component.

- 'App.test.js' is the unit test case for the default component.

viii)package.json

- information about dependencies(downloaded libraries)

======================================================

Component:-

======================================================

- Reusable part of a complex UI is called a component.

- As a react developer we can create more than one component.

- React applications are component based applications.

- Components are kept in the 'src' folder.

- Components are registered in the 'index.js' file.

Type of Components

- Functional Components

- Class Components

Class Component

- Simple JSX class can behave like component.

- Class components are created by extending the Component class.

- Component class is available in React class.

- React class is available in the react package.

- 'render()' is the mandatory lifecycle method in the class component.

======================================================

First Component

======================================================

directory structure

<>

src

first

- First.js

- index.js

\*\*\*First.js\*\*\*

import React from 'react'

export default class First extends React.Component {

render() {

return (

<div>

<h1> Welcome to First Component</h1>

<h3>React is Component Based</h3>

</div>

)

}

}

\*\*\*index.js\*\*\*

import React from 'react';

import ReactDOM from 'react-dom/client';

import './index.css';

import App from './App';

import reportWebVitals from './reportWebVitals';

import First from './first/First';

const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(

<>

<First/>

</>

);

// If you want to start measuring performance in your app, pass a function

// to log results (for example: reportWebVitals(console.log))

// or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals

reportWebVitals();

Directory Structure

<>

src

components

- MEAN.js

- MERN.js

- MEVN.js

- Fullstack.js

- index.js

\*\*\*MEAN.js\*\*\*

import React from "react";

export default class MEAN extends React.Component {

render() {

return (

<div>

<h1 style={{ color: 'blue' }}>Welcome to MEAN Stack</h1>

</div>

)

}

}

Similarly create MERN and MEVN Components

\*\*\*FullStack.js\*\*\*

import React from "react";

import MEAN from "./MEAN";

import MERN from "./MERN";

import MEVN from "./MEVN";

export default class Fullstack extends React.Component {

render() {

return (

<div>

<MEAN/>

<MERN/>

<MEVN/>

</div>

)

}

}

======================================================

State:-

======================================================

- State is used to store component data.

- {} used to display dynamic data.

Directory Structure

<>

src

Stateeg

- StateComponent.js

\*\*\*StateComponent.js\*\*\*

import React from 'react'

export default class StateComponent extends React.Component {

constructor() {

super()

this.state = {

data: 'Data from db Soon...!',

version: 18.2,

flag: true,

subs: ['ML', 'Maths', 'AI', 'IP', 'FSD'],

obj: {

fe: 'ReactJS',

be: 'NodeJS',

db: 'MongoDB'

},

products: [

{ "p\_id": 111, "p\_name": "P\_one", "p\_cost": 10000 },

{ "p\_id": 222, "p\_name": "P\_two", "p\_cost": 20000 },

{ "p\_id": 333, "p\_name": "P\_three", "p\_cost": 30000 },

{ "p\_id": 444, "p\_name": "P\_four", "p\_cost": 40000 },

{ "p\_id": 555, "p\_name": "P\_five", "p\_cost": 50000 }

]

}

}

render() {

return (

<div>

<h1>Welcome to State example</h1>

<p style={{ color: 'red' }}>{this.state.data} </p>

<p style={{ color: 'blue' }}>{this.state.version}</p>

<p style={{ color: 'green' }}>{JSON.stringify(this.state.flag)}</p>

<p style={{ color: 'maroon' }}>{this.state.subs} </p>

<p style={{ color: 'gray' }}>{JSON.stringify(this.state.obj)} </p>

<table cellPadding="5px"

cellSpacing="5px"

align='center'

border='1px'>

<thead>

<th>Sr no</th>

<th>P\_id</th>

<th>P\_name</th>

<th>P\_cost</th>

</thead>

<tbody>

{this.state.products.map((e, i) => (

<tr>

<td>{i + 1} </td>

<td>{e.p\_id} </td>

<td>{e.p\_name} </td>

<td>{e.p\_cost} </td>

</tr>

))}

</tbody>

</table>

</div>

)

}

}