6/1/25

Full stack

1) frontend
2) Backend
3) Database.

Mern stack - Mongodb, Express.js reactjs Nodejs
Mean stack - " " Angular.js "

Exclusive tags in html ⇒

- header
- footer
- article
- aside
- nav
- main
- audio, Vedio

=> js can be added client side and Serve side

=> powerful programming lang
    ↳ Equivalent to C, C++

=> React and Angular are popular framework.

=> make pages interactives.

1. <html>
<body>
<h1> AIML </h1>
</body>
</html>

Implement js

→ Same code
→ impor

1. Same code
- Use Script
- inside body

<Script>
Console.log ("AIML")
- Used out lin side body.

2. <html>
<body>
</body>
<script src="Bl.js">
</script>
</html>

3.
<script>
alert ("SAI")
</script>

4. Use more alert to display many times

5) Create Variables.

let, Var, Const

ex: Var num = 100;

<html>
<body>
</body>
<script>
Var a = 100.52
let b = 200
Const = 300
Console.log(a,b,c)
alert(a+" "+b" "+c" ")
</script>
</html>

Var → global or function Scope.
→ can be redeclared within the same scope

let → block scope, local Scope
→ Cannot be redeclared in the same Scope

Es6

=> Ecmascript.

=> extend features of javascript.

Day-1

1) why Console:

-> used for debugging and logging info.

-> monitor js code

-> Real-time testing.

Why important ?

-> Helps debugging

-> Improves code quality

-> Speeds up development.

2) Tailwind Css

-> Modern css framework

-> Use predefined utility -> | bg-blue-500 text-white
classes (padding, margin)

-> responsive design.

-> Use Utility Classes directly in html
without custom css writing.

ex :-

<button class="bg-blue-500 text-white"> click me </button>

Css :-

-> we will write the custom styles separate
Stylesheet.
4/1/25

Es6 :-

js
ecma script

[arrow function] - under es 6
 -> provide function expression.

Traditional way
 -> function functionname() {
 }

Arrow function:
 -> eg:-
 const add = (a, b) => a + b;
 -> will not have fun keyword.

```
const howareyou = () => {
    return 100;
};
```

document.getElementById("response").innerHtml = howareyou
(document -> Object, getElementById -> method, innerHtml -> property)

=> Its from Es6 for efficiency in terms of s

=> Increase readability.

=> func can be created without name
function

Q) Design Simple Calculator by getting 2 numbers as input. display addition, sub, product quotient remainder by creating individual arrow function for the same.

Q) Create an array by taking array size and array elements from user. extract all the perfect numbers and even prime numbers from the array.

Push → (add)

```
<html>
<script>
Var arr = [1, 2, 3]
alert (arr)
arr.push ('30')
arr.push (30,40,50);
alert (push)
</script>
</html>
```

Pop → (remove)

```
Var arr = [ ]
alert (arr)
arr.pop()
alert (pop)
```

unshift (add)

Shift → (remove)

```
Var arr = [10, 2, 3, 30, 100]
alert (arr)
arr.shift ()
alert (shift)
```

unshift (add)

```
Var arr = [ ]
alert (arr)
arr.unshift (111)
alert (unshift)
```

Splice

arr.splice (start, delecount, item)

program :- arr

```
Var arr = [1, 2, 3, 4, 5, 6]
arr.splice (2, 2, 99)
alert (arr)
```

Output :

1 2 3 4 5 6

1 2 99 5 6

Slice

```
Var arr = [1, 2, 3, 4, 5]
arr.slice (0, 2)
alert (arr)
```

Output :

1 2

Oops - Object oriented programming

Structure.

Example :- class → Bird

Object → peacock, Dove, Sparrow, Parrot.

Properties → Colors, wings, Size, leg weight.

behaviours → flying, eating, singing (methods)

Object inbuild methods :-

1. keys
2. methods

Js promise.

→ promise is a javascript Objal

→ 2 Stages.

- Resolve (success)
- Rejected (failure)

Note :-

1. Invoking a function
2. Call back

8/1/25

Scenario

Andhra's home — 5 km — Person A
3 km — Person B
10 km — Person C

Q) Write a promise called andhra-BP.

Distance andhra → 5000 → A, 3000 → B, 10000 → C

Promise inbulid methods.

When there is more than one promise in order to review them we can use promise inbulids methods according to the requirement.

methods :-

1.) Promise.all
2.) Promise.any
3) Promise.allsettled
4) Promise.race

→ Once it's ...
it will stop

Promise.any

→ Gives the shortest time provided.
→ Status should be, true.

Promise.allsettled.

→ Will display one among these three states.
i) fulfilled
ii) rejected
iii) Pending.

Promise.race

→ Works for status both true and false.

Usage :-

→ It is a library or framework of JavaScript

Ex Website :- Netflix and amazon.
Html Website :- Youtube. Wikipedia.

Two important folders in React.

1. Public
2. Src

Three imp files.

1. index.html
2. index.js
3. index.css

Note :- As of now don't touch index files
Note :- Initially do or write your co
in app.js

DOM (Document Object Model)

React follows - VDOM (Virtual Dom)

Once here unlike html. Once Dom
Created the changes are manipulations
what we do gets completed and only

→ In Web application everything is

each and every thing is called as Components.

Types :-

1. functional Component
2. class "

JSX

→ Writing html inside javascript

→ javascript XML

Props → where Components will have more and States.

Props

→ It wont change

ex :- name → TATA's, birla's

States

→ It changes or we can change it

ex :- water level in bottle

* Intial state (full)
* Updated state (half)
* Current state (empty)

→ Flipkart website

Home page

- Grocery
- Mobiles
- fashion

} → Components.

Mobiles :

→ Component name = mobile

Props : name, Version, price

State : Discount, stock a

Passing props between components.

React ho

```
const app = (props) => {
    return (
    );
}
```

React hooks:

- Ealier the (it) industry they were using class component, reason being state concep was not available with function components.

- Now hooks is used to implement state in function components.

Types :-

1. Use State => takes one argument (Initial State) => retains (array) (two values, Initial and update)
2. Use effect
3. UseRef
4. Use Context
5. Usereducder.

Example :- Counterlock

Starting the intial state as 0, we can increment, decrement, rest it using Use state Hook.

10/1/25

Program 1:

one : [ ]

two : [ ] , (Combine)

click : [ ]

Program 2 :- using hooks.

=> Array program without using hooks

=> " " with " "

=> Spread operator.

11/1/25

UseEffect

=> upon ~~under~~ the condition or action we apply in our functional components. monitoring the impact or side affects can be done use effect hook.

=> It accepts 2 arguments.

1. Callback
2. Dependency array (optional)

Note:

Call back function is like Constructor in java

20/1/25

Q) There are 5 components namely C_1, C_2, C_3, C_4, C_5 and add can display messages Components 1, 2, 3, 4, 5 respectively by keeping component 4 as parents. and rest all children, so C_1 child is C_2; C_2 child C_3, C_3 child is C_4. C_4 child is C_5 every component (it name as) should display message as C_1, C_2, C_3 .... display them side wise form h_1 to h_5

Note :- Whenever we use something inside {} it can be either js object or react Component.

Props can be passed between Components only by following the hierarchy which means: parent to child.

To Over come this in terms of efficiency we are using hooks.

Conclusion :-

If we want to use state from component to another component only way to achieve is passing it as props in the hierarchy.

=> This is not effiient.

=> TO make it efficient we have to execlusive hook called Use Context.

Q)

- Components
  - 1st => app.js
  - 2nd => Container
  - 3rd => Users
  - 4th => User

Use Context :-

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

- CreateContext
- Use Context

In the given example Createcontext will be in App Component and that will be used in User Component using Use Context

Use Effect :-

Synchronizing a Component with an external System.

After our action monitoring or checking (seeing) the side effects happening in the functional components. is possible using use Effect hook

3/10/25

Use Reducer :-

- Same as use state to manage or update states that is data that is value of Components.

Diff is if u have more states or Complex things u use use reducer hook.

Step 1 :-

Create increment decrement program. Using Use state

Step 2 :-

Replace Use state with Use Reducer

Arguments :-

1 → Reducer function (eg :- +, -)

2 → Initial value (use state)

↓ returns array with two values like Use state

=> first is initial count and second is dispatch function.

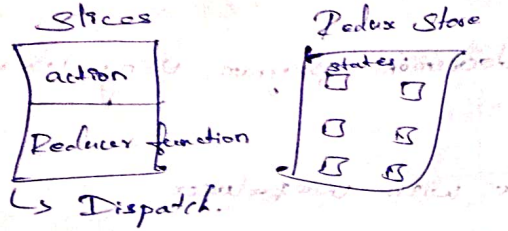
=> we can call them as (state, dispatch)

Here state will hold initial value and updated Once you call dispatch function and dispatch will trigger the use Reducer function.

Q) Get the password from ... as input. if password is correct display login granted: if the password is incorrect display Access denied

Component.

Redux :-



Install :

npm : @reduxjs/toolkit react-redux

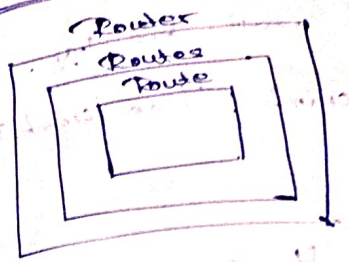
↓ library ↓ package

Connect react Components

To create redux store and slices

↓ js library

Router :-



Redux : store.js :-

Explanation :-

userInfo is the key for reducers and UseReducer is the name we give, for reducer action we get from UserSlice.js

After connection is done

Hook

:UseSelector hook - To Access Data from Redu

User.js

=> State . UserInfo . Users

↓ ↓ ↓

Reducer from store slice

Callback function

↓

key for reducer.

State is callback function.

→ 9/01/25 Bootstrap

→ npm : bootstrap react-bootstrap.

Button. Alert. Breadcrumb, Breadcrumb.Item

→ 10/01/25 Mongo db

→ Its No SQL in process to use the unstructured data;

eg :- Json file or data;

→ Json file will look like javascript object.

Compass :-

→ helps to fetch data from mongo db server. means compass helps us to reach mongo db like client.

→ It is an interface.

Mongosh :-

→ ~~Mongo~~ MongoDB Shell, gives you an interactive environment where you can run queries

→ Manage databases and perform tasks.

Two terms imp
Data modeling, Schema

DATA MODELING :

→ fix structure of your data, planning the structure.

eg :- employee details
name
id
password.

SCHEMA :-

→ actual blue print of your database which have you be created by fixing the format using data

eg :- employee details - id - int
name - string
salary - float

No SQL

SQL - Record

MONGO - Document.

SQL - Fixed Table

MONGO - collections → stored in DB

Mongo will have multiple DB

1. `Use dbname (emp)`.
2. `show dbs`
3. `db.emp.insertOne({ })`
   ↳ `empname:" "`
4. `db.emp.insertOne({empname:" ", Contact:" "})`
5. `db.emp.find()` → will show all the doc create.
6. `cls` → to delete the screen (not data)

→ Imbedded doc → (Object inside object)

7. `db.emp.insertOne({ favs:{ , }})`
8. `db.emp.insertMany([{ },{ },{ }])`
9. `db.emp.find({name:" "})`
   ↳ to search particular value
10. `db.emp.updateOne({name:" "},{$set:{age: }})`
11. `db.emp.updateMany({age: },{$set:{age: }})`
13. `db.dropDatabase()` → delete entire db.
14. `db.emp.find({hobbies:{$nin:[' ']}})`
   ↳ not existing
12. `db.myCollection.replaceOne({ },`
15. `db.emp.find({$and:[{price:{$gt: }},{Brand:' '}]})`
   ↳ display when both are true.
16. `db.emp.find({$or:[{price:{$lt: }},{Brand:' '}]})`
   ↳ display when any one is true.

1) Create a db "Computers"

2) One collection "laptops"

document field → name
- model
- color
- status → (available, not)
- price

One more object
- Vendor → Vname
- Vprice

Minimum 10 records.

List out particular model laptops

change its status to Avail ↔ Not

Q) filter all the records with the price range ~~15000 - 50000~~ range and go with dell and update with one more field shipping "s"

Q) id, name, age, city

DB - Computer.

Collection - Details.

Schema
- id number
- name String
- age no
- city String

Command:

db.customers.find({$and:[{hobbies:{$exists:false}},{age:{$gt:30}}]})

It will display where the hobbies field does not exist and age should be greater than 30.

Aggregation is group by
- Mongodb
- MySQL

db.Companies.aggregate([{$lookup:{from:'employee', localField:'_id', foreignField:'company_id', as:'Employee'}}])

=> local field - Employees
=> foreign " - Companies

New matched doc will be added as an array in Companies and the array name is employees.

Q) Create a DB - 'Bank'
create two collections - 'Customer' 'personal'
- 'Customer acc'

Data model for 1st Collection - name, address, phone
- name: String
- address: array
- phone: object (2 num)
- age

2nd " - accNo, Bank, branch
- accNo: int
- Bank: String
- branch: String
- IFSC: String
- current bal - float
- Acc type
- OD < Yes / No

Query:

1) filter only OD category Yes
2) Display only Customer Address where the branch name is same.
3) curr bal 10,000 - 20,000 filter their phone no.
4) filter only savings acc people
5) Add field "status" - same - to the people who have IFSC

Backend :-

→ Middleware

→ node.js

Node

→ Backend library from js.

→ In node we can use express.js as middleware.

To run : node filename.

```
const http = require('http');
const express = require('express');
const sayHello = require('./greet');
```

Node.js

Runtime environment that allows you to run js on the server-side

Note :

* Maintain split terminals (2) in order to use client and server.

Run Command :-

→ Start the server first always.

- node server.js

→ Command to run client

- npm start

1) To install express

-npm i express

"require" is a keyword in node.js

"get" - requesting from server

"post" - give info to server.

"listen" - to active the port

5) We can delete "package.json" to get it back the command.

→ npm init -y

Axios:-

- Its a popular js library used to make request from browser or node.js
- Axios is known for easy and clean syntax also flexibility especially works will with API and rest API's
- When we write API for exclusive purpose we call it is Rest API.

Cors:-

Cross Origin Resource Sharing.

When a webpage request information from resource [from other sites] whether to accept the request or proves it or not will be defined in a ~~note~~ rule book for this purpose we use cors:

In the given example we are requesting data from server

→ In DataComponent.js as client Using http get method via API.

Server response as json

from the json file I want to filter only msg so we are using ~~cors~~ from cors

response.data.message.

28/1/25

Steps:-

1) Inside src → Create 3 files.
   - Users.js
   - CreateUser.js
   - UpdateUser.js

Model.
- name
- age
- Email

Dependence

npm i react-router-dom

npm i axios

npm i express

⇒ In app.js add Routing for 3 Components.

⇒ Create a folder called server.js

Inside server path npm init

Index.js which is inside server folder write backend code.

Create another ~~connection~~. collection.

→ empl - emp-id
emp_name
emp_salary
contact [Array < one / two