6/1/25

1) Frontend
2) Backend
3) Database

MERN stack - MongoDB, Express.js react.js Node.js

MEAN stack - MongoDB Express.js Angular.js Node.js

Exclusive tags in html 5

- header
- footer
- article
- aside
- nav
- main
- audio
- video

Why html 5 ?

- Enhanced structures
- Multimedia support
- Mobile - friendly

⇒ js can be used as client-side and server side

⇒ js - powerful programming language

⇒ React and Angular are popular frameworks or libraries of js.

⇒ Make pages interactive/responsive

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 outside/inside body tag

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

3. <html>
<body>
</body>
<script>
alert ("Gaja")
</script>
</html>

4. Use more alert to display many boxes.

Create variable
let, var, const

ex: var num = 100

```
<html>
<body>
</body>
<script>
var num =100
Console.log
Var a = 100.56
let b = 200
Const c = 300
console. log(a, b, c)
alert (a+" "+b" "+c+" ")
</script>
</html>
```

Var : Function or global scope

let : Block scope, local scope

⇒ const var is intialized cannot be changed

1) Do example prgm for each of following:

i) simple if
ii) if else
iii) else if
iv) else if ladder
v) nested if

2)

Case 1:

No. of lemons in hand : 7

Expected output :

God 1 = 7 offered
god 2 = Need 7
god 3 = Need 7

Shortage !14

Case 2:

No. of lemons in hand : 21

Expected output :

God 1 = 7 offered
God 2 = 7 "
God 3 = 7

Sufficient

Case 3:

No. of lemons in hand : 15

Expected Output :

God 1 = 7 offered
God 2 = 7 Offered
God 3 = having 1 need 6

Shortage : 6

Case 1 :

No. of lemons in hand : 67

god 1 : 7 offered
god 2 : 7 offered
god 3 : 7 offered

surplus : 46

2) Sai is having 75 candies. Sai gives half of it to his friend gaya. Since Gaya loves sai a lot she gives back half of her portion. Calculate and display individually how many gaya or sai is having.

Constraints

→ Use one variable
→ Use function only one time without argument without return type.

$\frac{75}{2} = 37.5$ Angel.

$= \frac{1875}{100} =$ 18.75

37.5
18.75
56.25

⇒ 56.25

ES6

=> ECMASCRIPT

=> extend feature of javascript

Day-1

1) Why console:

→ used for debugging and logging information

- monitor js code.
- Real-time testing

Why important

→ Helps debugging
→ Improves code quality
→ Speeds up development
→ Enhance performance

Tailwind CSS

→ modern CSS framework
→ use predefined utility → (bg-blue-500, text-white, p-4)
classes (padding, margin, color..)
→ Responsive design

CSS

→ We will write the custom styles in separate stylesheet

Ex:

```
.button {
    background-color : blue;
    padding : 10px;
}
```

Tailwind CSS

- We use utility classes directly in HTML without custom CSS writing.

Ex:

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

11/25

ES6

⇒ extend feature of js

⇒ ECMASCRIPT

⇒ arrow function comes under ES6

Traditional func

→ function functionname() {

//code.

}

Arrow func

→ will not have func keyword

ex:

→ const ~~functionname~~ howareyou. = () ⇒ {

return 100;

};

document.getElementById("response").innerHTML = howareyou();

document → object

getElementById → method

innerHTML → property

⇒ Its from ES6 ⇒ for efficiency in terms of space

⇒ Increase readability

⇒ function can be created without name

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 & array elements from the user Extract all the perfect numbers and even prime no from the array.

2

6, 28

Push (add)

```
1. <html>
   <script>
   var arr = [1, 2, 3]
   alert (arr)
   arr. push (30)
   arr. push (100, 300)
   alert (arr)
   </script>
   </html>
```

Pop (remove)

```
var arr = [ ]
alert (arr)
~~alert~~
arr. pop ( )
```

1, 2, 3, 4, 5, 6

Shift (remove)

```
var arr = [1, 2, 3, 100, 300]
alert (arr)
arr. shift (~~0, 555~~)
alert (arr)
```

unshift (add)

```
arr. unshift (111)
alert (arr)
arr. unshift (111, 122)
alert (arr).
```

Splice 1, 2, 3, 4, 5, 6

splice (1, 99)

2, 2, 99

1, 99, 2, 3, 4.

1, 2, 99, 4, 5, 5

Splice:

arr.splice(start, deletecount, value)

Slice:

arr.slice(start, end)

OOPs

- Object oriented programming structure.

ex: - Class - Bird

Object - Peacock,
Sparrow,
Parrot,
Pigeon

Properties - Colors,
wings,
legs

Behaviours (methods) - flying,
eating,
singing

Object inbuilt Methods :-

1. keys
2. values
3. entries

Js Promise

- promise is a javascript object
- 2 stage
  - Resolve (Success)
  - Rejected. (Failure)

Note - 1. Invoking functions
2. Call backs - In one func calling another function.

7/1/25

Tailwind CSS

- utility-first css framework
- easy to build designs without writing custom css,
- Directly to HTML

We can use Tailwind CSS

- Using CDN link
- " package Manager like npm

8/1/25

Scenario 1

Andhra's home — 5 km — Person A

3 km — Person B

10 km — Person C

Write a promise called andhra - BP

Distance: Andhra — 5000 — A; 3000 — B; 1000 — C

setTimeOut (callback, delay)

↓

builtinfunction

Promise inbuilt methods:

- When there is more than one promise in order to preview them we can use promise inbuilt methods according to the requirement

Methods:

i) promise.all

ii) Promise.any

iii) promise.allsettled

iv) promise.race

Promise.all

- once it sees a promise is false it will stop

Promise.any

- gives the shortest time and provided status should be true.

Promise.allSettled

- will display one among these three states

i) fulfilled
ii) rejected
iii) pending.

Promise.race

- gives status can be true and false

React js

- Its a library or framework of Javascript

example websites:

- Netflix
- Amazon

Html website - Wikepedia, youtube

Steps:

1) node -v
2) npm -v
3) To create:
   npx create-react-app appname
4) npm start

important folders in React

- Public
- src

3 important files:

- index.html
- index.js
- index.css

Note : As of now dont touch index files.

Note : initially write your code in App.js

DOM (Document Object Model)

React follows - VDOM (Virtual DOM)

once here unlike html, 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

=> In web application, created by reactjs each and everything is called as components

2 components
- functional component
- class component

JSX:
- javascript XML
- writing HTML inside Javascript

9/1/25

Every component will have Props and States.

⇒ Props - It won't change

Ex: Name ⇒ TATA's, Bisleri

⇒ 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
- mobile
- fashion

} components

Mobiles

component name : Mobiles

props : name, version, price

state : offer, stock available

Passing props between components.

```
const App = (props) => {
    return (

    );
```

React Hooks

- Earlier in IT industry they were using class component. ~~reason~~ reason being state concept was not available with functional components
- Now hooks is used to implement state in functional component

Types
1. useState → 1 arg ⇒ Intial state
2. useEffect
3. useRef
4. usecontext
5. Usereducer

Ex:
CounterClock
Stating the intial state as 0, we can increment, decrement, reset it using useState Hook.

10/1/25

Prgm 1:

one - [1, 2, 3] — make this static array

two - [11, 12, 13]

Click - [1, 2, 3, 11, 12, 13]

Prgm 2

make it as a dynamic array.

=> Array prgm. without using Hook.

=> Array prgm using Hook

=> Spread operator

11/1/25

UseEffect

⇒ Upon the condition or action we apply in our functional components monitoring the impact or sideeffects can be done using useEffect Hook

⇒ It accepts 2 arguments

1. Callback function
2. Depedency array (optional)

Note: Call back function is like constructor in java.

20/1/25

Q: There are 5 components namely C1, C2, C3, C4, C5. and add and display messages components 1, 2, 3, 4, 5 respectively by keeping component C_1 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 should display its name as message as C1, C2, C3... display them side wise form h1 to h5.

Note:

Whenever we are using 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 overcome this in term of efficiency We are using hooks

Conclusion: If we want to use state from 1 comp to another component only way to achieve is passing it as props in the hierarchy.

~~UseContext Hook~~:

⇒ This is not efficient.

⇒ To make it efficient we have an exclusive hook called UseContext

Q:

⇒ 4 components
- App.js
- container
- Users
- User

UseContext :

Without following the Hierarchy passing state from one component to another component in an efficient way using this Hook.

→ Create Context
→ useContext

In the given example createContext will be done in App component and that will be used in user component useContext

UseEffect :

Synchronizing a component with an external system.

After action monitoring or checking (seeing) the side effects happening in the functional component is possible using useEffect Hook.

InnerHTML - display in screen.

used

useReducer

- Same as useState 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 useReducer hook.

Step-1 :

Create +, - prgm using useState.

Step-2 :

Replace with useReducer

⇒ Takes 2 arg

1 → reducerFunction (ex: +, -)

2 → Intial value of state

⇒ Returns an array with 2 value

⇒ can use for multiple values

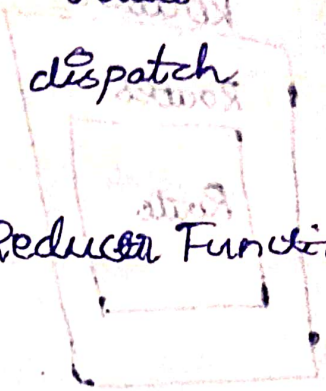
first — intial count

second — dispatch (*)

We call them as (state, dispatch) like (theme, setTheme).

Here state will hold intial value and updated once you call dispatch. function and

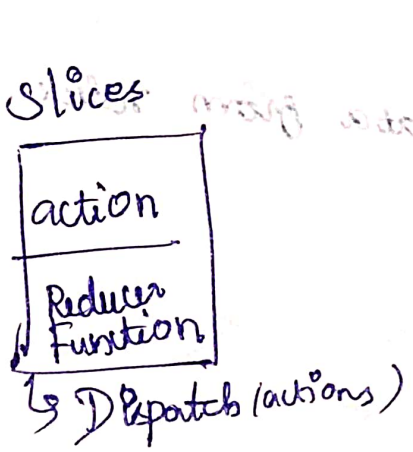
Dispatch will trigger useReducer Function

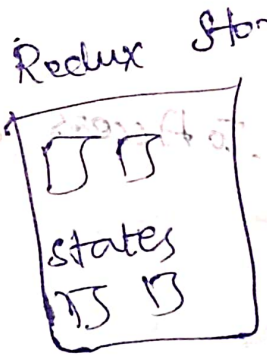


Q:

Get password as input from the user, if password is correct display component Login granted, if pw incorrect Display Acess Denied component

Redux:





Install - npm ; @reduxjs/toolkit ↓ react-redux

Library ↓ Package

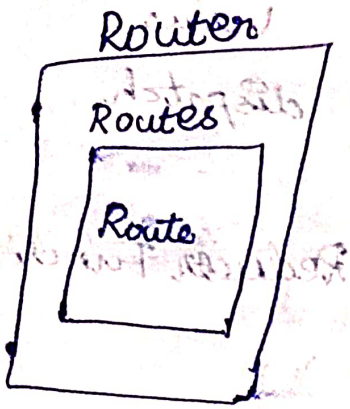
Connects react components

To create redux store and slices

↓

js library

Architecture :



Redux

Store.js

=> userInfo is the key for reducers and userReducer is the name we give for reducer actions we get from userSlice.js

After connection

Hook

useSelector hook - To Access data from redux

users.js

=> state . userInfo . users

state → reducer Callback func

userInfo → from store → key for reducer

users → slice

Here

State is callback function

Hook - 2

useDispatch

Bootstrap

⇒ npm bootstrap react-bootstrap.

Button, Alerts, Breadcrumb Breadcrumb.Item,

Card

↓

Container

⇒ Card

MongoDB

- Its NOSQL

↓

Process unstructure data

ex: JSON data

JSON will look like Javascript object.

Two terms imp ⇒ Data modeling, Schema

Compass :

→ helps to fetch data from mongodb sever

→ Its an interface

→ helps to reach mongodb sever

Mongosh

⇒ MongoDB: shell is replaced with Mongosh

⇒ interactive environment where you can run queries, manage db and perform tasks.

Data Modeling

Fix the structure of your data

Planning the structure

ex: employee details:
- name
- id
- pass

Schema

⇒ actual blue print of your database by fixing the format using Data Modeling.

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

---

collections
↓
Database.

NoSQL

SQL	MongoDB
Record	Document
Table	Collection ⇓ stored in DB

Mongo will have multiple DB.

MongoDB Commands:

1) use db_name
2) show dbs
3) db.emp.insertOne({empname:" "})
4) db.emp.insertOne({empname:" ", id:" "})
5) db.emp.find() → will show all the dbs created
6) cls – to delete the screen (not data)
7) embedded document → object inside object.
   db.emp.insertOne({ , fays:{ }})
8) db.emp.insertMany([{ }, { }, { }])
9) db.emp.find({name:" "})
   ↳ to search particular value
10) db.orders.drop() → delete content
11) db.emp.updateOne({ }, {$set:{ }})

1) Create a db "Computers"

one collection → Laptops

documents ⇒ name
model
colour
States ← Available / Not
price

One more object

Vendor :

name
Vprice

Minimum 10 records

List out particular model laptops

Change its status to Avail — Not

---

12) db.myCollection.replaceOne (

{ name: " " },

{ name: " ", age: 32 })

13) db.dropDatabase() → delete entire db

14) db.emp.find({hobbies: {$in: ['']}})

not existing

Q: Filter all the records with the price range 30000 - 50000 range and go with dell and update with one more Field "Shipping: 'S'"

Q: id
name
age
city

Computers — db
Details — collection

Schema
id — number
name — String
age — no
city — string

1)

db.products.find({$and:[{price:{$gt:30000}},{brand:'Apple'}]})

It will display when both are true.

2)

db.products.find({$or:[{price:{$lt:10000}},{brand:'Apple'}]})

It will display any one of the true value.

3) db.customers.find({$and:[{hobbies:{$exists:false}},{age:{$gt:40}}]}).

⇓

It will display where the hobby field does not exist and age is greater than 40.

4) aggregation is GroupBy

⇓ ↓

MongoDB. InMySQL

5)

```
db.companies.aggregate([{$lookup: { from: 'employees',
localField : '_id', foreignField : 'company_id',
as : 'Employees' }}])
```

⇒ Local field - employees
⇒ Foreign field - companies

New matched doc will added as an array in companies with the name Employees.

Q:

Create a DB "Bank"
Create 2 collections — "Customer Personal", "Customer Account"

Data Modeling for C1 — name - string
address - array
PhoneNo - object
(2 Num)
1 2
Age - NO

C2 — AccNO - int
Branch - string
id
BankName - string
IFSC code - string no
Curr bal - float
Acc type - saving/curr
OD - yes/no

Query

1) filter - only : OD category. Yes

2) display only customer Address & where the branch name same.

3) Curr bal 10,000 - 20,000 filter their Phone No.

4) filter only the savings account people

5) Add Field " Status : Same "

||

IFSC code same records

Node

→ backend library from js

→ In node we can use express.js as middleware

To run use: node filename

```
const http = require('http'); // Built-in module
const express = require('express'); // Third-party module
const sayHello = require('./greet'); // Custom module
```

Node.js

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

⇒ Library or framework

Note: Maintain split terminals (2) in VS - code in order to use client and server

Run command :

1. Start the server first always
   = node server.js
2. To run the client
   = npm start

1) To install express

- npm i express

Note: "require" is a keyword in node.js

http methods: "get" - requesting from sever
"post" - give info to sever

"listen" - to activate the Port

2) We can delete package.json to get it back the command is:

node ⇒ npm init -y

Axios:

- It is a popular js library used to make http request from browser or nodejs

- Axios is known for easy and clean syntax also flexibility especially works well with API and rest API's

When we write API for exclusive purpose — we call it as Rest API.

Cors

Cross Origin Resource Sharing

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

axios — npm i axios

cors — npm i cors

Datacomponent and Server example

In given eg we are requesting data from server which is hello this is a data from sever.

In DataCompent.js as client using http get method via API/data

Server response as JSON

From the Json file I want to filter only the msg, so we are using from cors response.data.message

1) Create 3 files inside src

=> Users.js

=> Createuser.js

=> Updateuser.js

Model - name
age
email

Dependencies

npm : react-router-dom

npm : axios

npm : express

npm : cors

In App.js

-> Bring routing for 3 components

Create folder server

Inside server path npm init

Index.js which is inside server folder write backend code

Create another collection inside same db

emp1 - empid
- empname
- Salary
- contact (Array)