6/1/25

Why HTML?

defines structure and content of web pages, enabling browsers to display text, img. serves as foundation of web develop... & ensures content is accessible across devices and platform.

HTML5 tags?

<header> <article> <aside>
<footer> <sections> <nav>

→ JS can be used as client and server side

→ Powerful programming lang apart from being used in Frontend, be, equivalent to C, C++ and Python.

React and angular js popular frameworks or libraries of js.

code for creating alert box displaying

```
<html>
<body>
<h1>AIML</h1>
<script>
var a = 100
var b = 200
const c = 300
alert(a, b, c)
console.log(a, b, c)
alert(" "+a+" "+b+" "+c)
</script>
</body>
</html>
```

var allows redeclaration,

-> let & const don't allow reassignment

-> let ,, ,, block-scoped

var () scoped.

let - we cant print before initialising.

Pgm diff btw var & let using ().

```
<html>          <script>        {if(x==100){
<body>          var x=100;        var a=10;
</body>         funation test()   var b=20;
                                  console.log(a);
                                  console.log(b);
                                  } test();</script>
                                  </html>
```

1) DO example programs for each of the following

i) simple if
ii) if else
iii) else if
iv) else if ladder
v) Nested if
vi) for loop
vii) while loop
viii) do while
ix) switch case

Lemons 3 GODS

case 1:

No of lemons in hand: 7.

Expected output:

God 1: 7 offered

God 2: Need 7

God 3: Need 7

Case 2:

No of lemons in hand: 21

god 1: 7 offered

god 2: 7 offered

god 3: 7 offered

sufficient

case 3
No of lemons in hand 15
god 1: 7 offered
god 2: 7 offered
god 3: having 1 need 6
shortage: 6

case 4
No of lemons in hand: 69
god 1: 7 offered
god 2: 7 offered
god 3: 7 offered
surplus: 46

Day 2

7/1/25

In Git daywise create a folder, 6 Jan 2025 No space

Into 2 folders
1) content
2) handson

Tailwind css

we can track program flow = console
~~debug~~

function 4 types

ES6
called ecma script
arrow fun - under ES6
obj : method

Arrow fun:

Its from es 6 for efficiency in terms of space and to ↑ readability we can create functions without name and its called as arrow function.

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

- document → js object
- innerHTML → Property
- getElementById → Method
- "response" → Id name

element is HTML element

Design simple calculator by getting two num as input +, -, x, Quotient, Remainder by creating individual arrow functions for the same.

Create an arr by taking arr size & arr elements from the user. Extract all the perfect numbers and even prime numbers from the array.

Arr: <html>
<sc..>
var names = ["...."]
names[5] = "rider"
alert(names)
</sc..>
</html>

→ This printed in chrome alert box

shift used to remove
unshift used to add

OOPS

object oriented Programming structures

eg: class
bird ∧ class have multiple objects
objects

class =
Bird = Peakock
Properties: colour, size, weight
Behaviour: (Methods)
↓
Flying, eating

JS Promise:
JS object. 2 states for promise
1) Resolved (success)
2) Rejected (failed)

NOTE
1) Invoking a function
2) callback

Tailwind CSS
- utility - first CSS framework
- easy to build designs
writing custom css
- Directly to HTML

8/1/25

Write a promise called Andhra - BP

Distance = Andhra to A = 5000

Andhra to B = 2000

Andhra to C = 1000

1st Reach, B next, A next

```
<html>
<script>
let reachA = new Promise((resolve, reject) => {
  let reached = true
  if(reached)
  setTimeout(resolve, 1000, "A reached")
  inbuilt fun
  else
   reject("A not reached")
})
```

2 parameters

to call if promise is success

Promise represent value that may be available now or future or never

→ Boolean var indicating destination (A) reached or not

Promise Inbuilt Methods:

when there is more than one promise involved, in order to resolve them we can use promise inbuilt methods according to the requirement.

METHODS

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

Promise.all([reachA, reachB, reachC])
.then((message) => console.log(message))
.catch((message) => console.log(message))

handle resolution & rejection of Promises
in JS:
→ define what should happen when promise is fulfilled
→ when promise is rejected.

Promise.all

once it sees a promise false it will stop

Promise.any

Gives the shortest time promise provided. status should be true.

Promise.allsettle

will display one among these three states or results.

1) Fulfilled ⇒ approve
2) Rejected ⇒ Not approve
3) Pending

```
function one(){
    var x=100;
function two(){
 console.log(x)
}
two()
}
one()
```

React JS:

It is library or Framework of JS

ex:

Netflix, Amazon

HTML websites = Youtube, Wikipedia

---

steps:

1) node -v

2) npm -v

3) npx create-react-app demo appname

4) npm start

cd demo

code .

1. create a folder called react in desktop
2. double click folder - address bar (cmd)
3. check the version

commands:

1) node -v (check version of node)
2) npm -v (check version of npm)

4) creating React app using below command
npx create-react-app demo

5) npm start

Readport 3000
angular 5000

make it work

1) open your react folder
u will see ur app
folder got created

to open my app in vs code

1) go to and promt
2) cd demo
3) code .

terminal => new terminal

npm start

npm i web-vitals

React follows

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

npm install -g npm@latest

Web application:

In WA created by react js each and everything is called as component.

Types of components:

1) Functional component
2) class component

9/1/25

Props and states:

Properties:

⇒ Every component will have props and states.

Props:

If won't change: eg. name Tata's Bisleri.

States:

It changes or we can change it.

eg: water bottle level; water level in bottle.

Initial state (full)

updated state (half)

current state (empty)

Flipkart website

Homepage

Grocery
Mobile
Fashions
} components

Mobiles

component name: Mobiles
Props = name, version, Price
state → discount

Passing Props between components

⇒ create new files

using arrow fun child with props
Access property called message.

React hooks:

Earlier in IT industry they were using class component. Recent being state component was not available with functional component.

Hooks is used to implement state in functional component.

Types:

1) use state
2) use Effect
3) use Ref
4) use context
5) use Reduce

Best ex for use state: counter clock.
Starting the initial state as zero we can incremented decremented resetted using use state hook.

Spread operator:

```
//Initial arrays
const array1 = [1, 2, 3];
const array2 = [4, 5, 6];
Const combinedArray = [...array1, ...array2];
console.log('Array1:', array1);
Console.log('Array2:', array2);
Console.log('combined Array:', combinedArray);
```

Passing arr into usestate

COMPONENTS INTERCOMMUNICATION:

create Parentcomponent, childcomponent. Parentcomponent will pass data and a function to the childcomponent, and the childcomponent will use the function to send data back to the Parentcomponent.

10/1/25

Pgm 1

one - [1, 2, 3] make static array
two - [11, 12, 18]
click - [1, 2, 3, 11, 12, 18]

Pgm 2

make dynamic array.

-> Arr prgm with & without hook

Parent component:

Data from Parent : Hello from Parent

Data from child : Hello from child

child component

Data from Parent : Hello from Parent

Send Data to Parent

Shopping Cart

Items in cart:

Product list

- Laptop [Add to cart]
- Smartphone [Add to cart]
- Headphones [Add to cart]

11/1/25

Use effect - 1

Component name = Timer

↳ upon the condition or ~~connection~~ Action we apply in our functional components monitoring the impact or side effects can be done using use effect hook.

use effect accepts 2 arguments

1) call back function

2) Dependency array

NOTE: Callback fun is like constructor in java

MERN STACK

Props = cannot change

state = can change

↓

also knows as Data

ex: counter clock

Usestate Hook = To update the state

↳ Returns ↓

an arrays very 1st Hook

→ arr of 2 val

Usestate → it takes 1 arg ↓ that is initial state passed as arg

→ Initial state ⇒ what

→ Update state

↳ It's a function ⇓ To update state

another Hook

Useeffect

Before Page

eg: cbe to Pollachi

Time & space save in web application ⇓ run time increases so save time.

state can pass as prop

prop to state but hierarchical

In react everything is component

In comp ⇒ data

20/11/25

There are 5 components and add & display messages components, comp2, 3, 4, 5 respectively by keeping comp1 as parent. C1 is parent & rest all children as per the order. So C1 child is C2, C2 is child of C3, C3's child is C4, C4 child is C5. Every component will have or should display its name as a message. As component1, component2 till component5 and display them sidewise from H1 to H5.

C1, C2, C3, C4, C5

↑ Hi is a props pass to C5

create a folder React full

create app inside the folder

(Rafc)

Sometimes direct export don't work so use traditional export = export default

→ change uppercase (export component name).

→ Inside

<h1>COMPONENT 1</h1>

In C1

import React

import {C2} from './C2';

Next take C1:

after h1

<C2/>

↓

after

Putting child you should import

If wherever we are using something inside {} it can be either JS object or React component

App.js

return (

<C1 />

C1.js

expot const C1 = (props) => {

Props can be passed to component only by following the hierarchy which means parent to child

To overcome this in terms of efficiency we are using hooks

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

This is not efficient to make it efficient we have an exclusive hook called use Context

four component

1) app.js
2) container
2) users
4) use

Use context:

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

2 x things

1. create context
2. use context

In the given example create context will be done in app component and use that will be used in user component using use context.

clock.js -- clock

import { useState } from 'react';

function clock() {

-> functional → class = create

2 component ↓ = class

event

API = Fetch data from server

useReducer:

Same as usestate to manage or update states that is data that is values of components - diff is if u hv more states or complex things u use usereducer hook.

Step 1: Create Increment decrement program using usestate.

Step 2: replace usestate with use reducer

Note:

point 1: usereducer takes 2 arguments first is reducerfunction which says what u want to do (like increment or decrement), 2nd is initial value of state

Point 2: usereducer returns array with 2 values like usestate

1st is initial count & second is dispatch function. we call them as state and dispatch.

--- how this state will hold that initial value and updated once a call dispatch reducerfunction will be triggered.

this ex increment for both + & - bcs we wrote like this.

Write a program usereducer to print rainbow colours

1) Reset
2) speed up

App name = Redux.js

1) Display 2 static var val on screen
   ∧ should be js object

x : num
y : string

function jsobdisplay(){
var obj = {
num: 100
name: "Jith"
}
return(

<h1>{obj.number}</h1>
<h1>{obj.name}</h1>

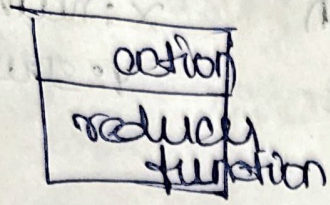
)}

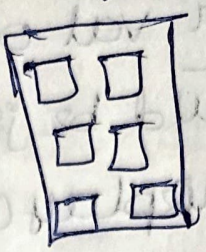
2) Get the password from the user as input if the pass is correct display the component login granted if pass is incorrect display access denied component

3) How to extract values from form

import

Redux





global state management

dispatch(action)

dispatch we give action that triggers reducer function which choose ~~from~~ state from store and come that is gives us data becoming more complicate

Instalation:

1) npm i @reduxjs/toolkit react-redux

@reduxjs/toolkit → redux library

react-redux → package

To create redux store ~~lib~~ created a library

2) npm i react-router-dom

add form in home.js

label: Name

input box type: text

add one button called "Add"

Create 2 folder inside src 1. app for store 2. slices for slices.

store = will have states

slices will have 2 things actions & reducers.

Step 2: Inside slices folder create file called userslice.js

import {createSlice} from '@reduxjs/toolkit';

To access data from redux use

state is reducer userinfo from store

users from slice

UseDispatch = It's a hook

Initiate we want to data or state

use usestate hook

onchange event

↓

output

name call

s

sa

sam

empty string to complete word

Ex:

```
FUNCTION HOME(){
const [name, setname] = usestate("")
function handlechange(event){
setname(event.target.value)
console.log(name)
}
<label>name</label>
<input type="text" onChange={handlechange}
value={name}></input>
<h1>{name}</h1>
}
```

work only for one ~~text~~ field.

In app.js

after <></>

import should be done

file name then folder name

```
import Form from './slices/Form';
```

file name

folder name

Bootstrap

npm install bootstrap react-bootstrap

MongoDB

Its NOSQL

ex: JSON data

Compass:

=> helps to fetch data

=> Interface

Mongosh

=> MongoDB shell replaced with Mongosh

Data Modelling

Fix struc of data

ex: name
id
pass

Schema

actual blue print of your database

NOSQL

SQL	MongoDB
Record	Document

Commands:

1) use dbname
2) show dbs
3) db.emp.insertOne({empname:"eg"})
4) " ({" ", Rat:"eg"})
5) db.emp.find()
6) ds - To delete

7) embedded document

insert many

8) db.emp.insertMany

9) " .find

10) db.order.drop()

11) db.emp.updateOne.

Qn: Filter records with price range 30000 - 50000 range & go will add & update one more Field shipping

1) db.products.find({price:{$gt:20000}})

display both true

2) db.prod.find({$or:[{price:{$lt:100000}},{brand:'Apple'}]})

display any one

⇒ local field - employees

⇒ foreign field - companies

Query

1) filter only OD category
2) display only custom address
3) cur bal 10,000 - 20,000
4) filter savings acc
5) Add field "status: Same"

Node

backend lib from js

To run node filename

Node.js

runtime environment
library

Run command:

⇒ node server.js

⇒ npm start

Axios:

- popular js lib used to make http req from browser.

→ known for clean syntax

Cors

Cross origin Resource sharing.

Axios - npm i axios

Cors - " cors

28/1/25

Create 3 files

→ Users.js

→ CreateUser.js

→ UpdateUser.js

Dependencies:

npm i react-router-dom

npm i axios

" express

" cors

Create another collection inside same db

Emp1 - empid

- empname

- salary

- contact(Array)