①:

```
<script>
function calc (candi) {
var c = 75
return (c-(c/4)) +(c/4)
}
var res = calc()
console.log(res)
</script>
```

②:

```
<script>
function calc(candies)
{
console.log("sam: " +
(candies - (candies/4))
+ " candies\n" + "angle
+ (candies/4) + "
candies");}
calc(75)
</script>
```

③:

```
<script>
function calc(candies){
return("sam" + (candies-(candies/4)) + "candies\n
+ "angle: " + (candies/4)+" candies");
var res = calc(75)
console.log(res)
</script>
```

ES6 is an extended feature of javascript known as ECMA script.

Why is console important?

⇒ used for debugging and logging information and also monitors js code.

It improves the quality of code, speeds up the development, and enhances its performance.

Tailwind CSS:

⇒ It is a modern css framework. It uses predefined utility. It gives responsive design.

CSS:

write code in custom styles in seperate stylesheet.

Eg: button {
background-color: green;
padding: 10px; }

7/1/25:

GIT:

create a folder day-wise

① 6Jan2025

again 2 folders: content / handson

Assignment: Tailwind CSS

ES6 is an entented feature of js.

Also known as ecma script.

The arrow function also comes under ES6

```
<html>
<body>
  <h1 id=" response"></h1>
  <h1 id = " res"></h1>
</body>
<script>
const howareyou = () => {
  return 100;
};
var add = (a, b) => {return a+b}
document.getElementById("response").innerHTML = howareyou();
document.getElementById("res").innerHTML = add(100,200);
```

funcs => methods
C => procedural lang
OOPS, Python => OOPS lang
HTML element | JS object

getElementById: method; HTML element; id name
innerHTML: property; displays ↑ OP; howareyou(): variable
js obj

ARROW FUNCTIONS:
It increases the readability, used for efficiency, we can create func without names and its called as arrow func.

Design a calc by getting 2 nos as input. Display add, sub, product, quotient, remainder by creating individual arrow functions for the same.

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

shift => delete
unshift => add

OOPS - Object Oriented Programming structure

Eg:
class - Bird
One class have multiple obj
Obj - Parrot, pigeon, peacock
Behaviour (methods)
flying, chirping, eating

JS promise:
promise is a js object.
2 status of promise: Resolved (success)
Rejected (failure)

① Invoking a function
② Callbacks

8/1/25 Write a promise called "andhra-bp"

Distance:
Andhra - A = 5000
Andhra - B = 2000
Andhra - C = 18000

EO:
C reached
B reached
A reached

In-built methods:
When there is more than one promise, in order to review them, we can use promise in-built methods according to the requirements.

Subtitle methods:
- promise.all
- promise.any
- promise.allSettled
- promise.race

promise.all => once it sees a promise false, it will stop.

promise.any => gives the shortest time promise provided status should be true

will display one among these 3 states : allSettled
⇒ fulfilled ⇒ pending
⇒ rejected

any ⇒ works only with true
race ⇒ even works with false

React ⇒ NETFLIX
Framework or library of javascript
eg: netflix, amazon

HTML : youtube, TCS, wikipedia

2 Imp folders in react:
① Public folder ② src folder

3 Imp files
① Index.html ② index.js ③ index.css
Note : as of now, dont touch index files
Initially, write ur code in app.js

DOM :

Html, css
React follows Virtual DOM (V-DOM)
Once here unlike HTML, DOM gets created.
the changes are manipulation what we do gets completed and only that part will be rerendered.

In HTML, everytime we make change, entire DOM will be re-rendered.

In web app created by reactjs, each and everything is called as components. Types : ① functional comp
② Class comp

---

writing html in js code

9/1/25 Props and States:

Every component will have props & states,

PROPS:
It wont change. Eg: name
TATA'S BISLERI

STATES:
It changes or we can change it
Eg: Water level in bottle
Initial State : Full
updated state : Half
current state : Empty

Flipkart website : Home page
Grocery
Mobile
Fashion
} components

Mobiles:
component name
props
version
price
State: discount

PASSING PROPS BW COMPONENTS:
*create 2 files
⇒parent.js / child.js

*open child.js
create a component called with the help of props using arrow function.
• enter rafc

state ->

earlier in IT, they
Reason: state concept was not available with functional components.

Hooks is used to implement state in functional components.

types: Types => useState | useReducer
useEffect
useRef
useContent

Eg: useState
=> counter clock
=> Stating the initial state as 0
we can increment / decrement / reset

spread operator:
(i) passing array in useState

useEffect - 1:
create timer.js
upon the condition or action, we apply in our functional components, monitoring the impact or side effects can be done using useEffect - hook.

It accepts 2 args: callback function
dependency array

callback func is like constructor in java.

Usestate takes 1 arg: ~~default~~ initial state
returns an array of 2 values: initial state
updated state

I have 5 components : c1, c2, c3, c4, c5
and add messages and display those c1, c2, c3, c4, c5 respectively. By keeping c1 as parent and rest all children as per the order. So, c1's child is c2, c2's child is c3, c3's child is c4, c4's child is c5
Every component should display a msg ⇒ its name as component 1, 2, etc. Display them size wise from h1 to h5

Traditional way of export ⇒ export default

curly braces can be used either in javascript object or react component

Props can be passed b/w components only by following the hierarchy which means parent to child.

To overcome this in terms of efficiency, we are using hooks.

Use ~~contacts~~ context hook :

conclusion :

If we want to use one component to other component, the only way to achieve it is by passing the hierarchy.

This is not efficient to make it efficient, we have an exclusive hook called ~~use~~ useContent.

1st cmp : app.js
2nd cmp : container
3rd cmp : users
4th cmp : user

without following hierarchy passing state rom one cmp to another cmp in an fficient way using this hook.

Subtitle: 2 imp things:
create content
use content

In the given example, create content will be done in app component and that will be used in user component using useContent

Title: useEffect:
Synchronizing a component with an external system
After our action, monitoring or checking the side effects happening in the functional (seeing) component is possible using useEffect hook.

Takes 2 arg: callback func
dependency array (optional)

USEREDUCER :

same as usestate to manage or update states that is data which is value of cmp.
If you have more states / cmplx things, use usereducer hook

S1 : create inc/dec prgm using useState
S2 : replace usestate with useReducer

Note :

Point 1 : useReducer takes 2 arg first is reducer function

second is ⇒ initial value of state

P2: Returns an array with 2 values.

1st is ⇒ initial count
2nd ⇒ dispatch function
We call them as state and dispatch
state will hold initial value and update it once you call dispatch function.
Dispatch will trigger useReducer function

Get a pw from the user as input
If the pw is correct, display "LOGIN granted"
If the pw is incorrect, display " ACCESS denied.

REDUX ⇒ Global state management

state ⇒ data

Slices

Action ⇒ updation of state | reducer function

Stores

States of the components

To create
INSTALL ROUTER : npm i react

Architecture of route :

router
routes
route
path
route
routes
router

userInfo is the key for reducers but useReducer is the name we give for reducer actions we get from userslice.js

After connection is done,

state is reducer userinfo from the store
from slice
const users = useSelector((State) => State.userinfo.users);

BOOTSTRAP

npm install bootstrap react-bootstrap

MONGODB :

JSON ; we use NoSQL to process unstructured data
It is a js object : Eg => json, json file
Js obj and json are not same

Compass : Helps to fetch data from mongodb server ; helps to reach u to mongo db as its client

MONGOSH :

It is an interactive environment.
Mongoshell is replaced with MONGOSH
We can run queries, manage db, perfo
administrative tasks

DATA MODELLING:
fixes the structure of ur data by planning the structure.

Schema:
Actual blueprint of db which we created by fixing the format using data modelling.

empdetails(name, id, salary)

NO SQL
SQL - RECORD
MONGO - DOCUMENT
SQL - TABLE
MONGO - COLLECTION

collections are stored in DB
mongo also will have multiple db

test > use aiml

db.emp.insertOne({empname: "name"})

1] create a db => computers
2] one collection => laptops
↳ name, model, color, status, price, [vendor
↳ vname
vprice]
3] status : available
not available

create minimum 10 records
1] particular model laps should be listed out
2] change its status to available -> new

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

and ⇒ shows the result if both are true
or ⇒ any one is true

all docs in the customer collection where the hobbies field does not exist and the age field is greater than 40.

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

The local field belongs to employees and foreign field companies.
Newly matched doc will be added as array in companies; new array is employees.

---

create a db called bank
create 2 collections under that
1st coll name is customer-personal
2nd coll name is customer-account

Data model: 1st coll
name, address, phNo, age
str array obj

Data model: 2nd coll
AccNo, branch, bankName, IFSCcode, curBal, accType,
int str str str str

* Filter only OD category
* Display only customer's address where the branch names are same
* curr_bal 10K-20K, filters only PhNos
* Filter only the savings account people
* Status same ⇒ add it when the IFSC codes are same

BACKEND

Node is a backend lib from js.
In node, u can use express js as middleware.

require from node.js - it helps to include modules in our code.

below is the eg of custom module

```
const http = require('http'); // Built-in
const express = require('express') // 3rd party module
// const
```

Purpose of node:

nodejs: It is a popular js library runtime environment which allows us to run js on the server-side

It maintain split terminals (2) in vs code in order to use client and server.

Run command:

start the server first.

cmd: node server.js

cmd to run client: npm start

require is
express() invokes it
"get" - request from the server
"post" - give info to server
"listen" - activate the post

node we can delete package.json, to get it back, the cmd: npm init -y is used
server's console is at the terminal

Axios:
Popular js library used to make http request from the browser or node js. It is popular because code is simple. known for easy & clean syntax; also flexibility especially it works well with API's and rest API's.
restAPI - closed network for a particular when we write API for an exclusive purpose, we call it as rest API's.

Cors:
cros origin resource sharing
When a webpage requests info from resource, (from other sites), whether to accept the req and process it or not will be defined in a rulebook, for this purpose we use cors.

npm i axios
npm i cors

In the given eg, we are requesting data from server.

hello this is data from server

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

Server responds as json

From the json file, I want to filter only the message. So, we are using from cors response ~~.js~~ data.message.

mangocrud

create 3 files inside src

Users.js
Createusers.js
Updateusers.js

~~Data~~ for all these

model
name
age
email

Button => AddUser

we need router, axios, source express(cors)

In App.js, bring routing for all these files

create a folder "server"

In index.js which is inside server folder, write backend code

create
database
emp
emp

Create a new collection under the same database "crud"

emp1 => model, schema
emp id, emp name, salary, contacts (array)

Create a new collection under the same database "crud"

emp1 => model, schema
emp id, emp name, salary, contacts (array)