06/1/2025

Full Stack development

Exclusive tags present under HTML5 :-

article, aside, figcaption, figure, header, footer, section, summary, time, details, menuitem

Why HTML 5 :-

* Enhancing code structure
* Cross platform compatibility
  ⟶ Device independence
* Offline support.

Day 1 :

Complete all the three assignments
(HTML5 tags, CSS

Tailwind CSS

Javascript

* Both
  - → client side (front end)
  - → Service side (connecting with database)

* JS can be used at client side and Service side.

* JS is one of the powerful language apart from being used in frontend also equal to c, c++, python.

Two terms:

* React
* Angular } → Popular framework or libraries of js

Console : y console :?

```
<html>
<body>

</body>
 <Script>
const c=187
Var num=100
let b=177
Console.log(num) ⟶ to see in console.
</Script>
</html>
```

* alert (num + " " + b + " " + c + " ")

Diff b/n var + let

- var → * function scope
- let → * Block scope

* Global

Creating function in js ⟶ function

① 100 examples programs for each of the following:

① Simple if
② if else
③ else if
④ else if ladder
⑤ Nested if.

Ex: Input : no. of lemons = 7

case :1

G1 = n = 7 offered

G2 = n = 7 needed.

G3 = " "

Shortage = 14.

case : 2

lemons in hand = 21

Grod 1 = 7 offered

G 2 = 7 "

G3 = " "

Sufficient

case 3: lemons in hand is 75

G1 = 7 offered

G2 = " "

G3 = having 1 need 6

75
37.5 56.25
18.75
18.75

case 4: lemons in hand 67

G1 = 7 offe

G2 = "

G3 = "

Surplus : 46

qn:

Sam is having 75 candy. He gives Half of it to his friend Angel. Since, Angel loves Sam a lot She gives back half of the portion. calculate and display individually how much Angel or Sam is having?

Constraints : ✱ Single variable if possible ~~only~~ one line

✱ use function.

Array:
* Extended Version of JS.
* Single variable has many values

1) Why console:
→ used for debugging and logging information.
→ Monitor js code

Why important:
- helps to debugging
- Improves code quality
- Speeds up development
- Enhance performance.

Tailwind css:
→ Modern css framework
→ Uses predefined utility.
→ Responsive design.

css:
→ we will write the custom styles in seperate stylesheet.

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

Tailwind css:
- use utility classes directly in HTML without custom css writing.

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

6/jan/2025 → Content (notes)
→ Handson (Programs / assignments)

07/05
Assignment
Tailwind

Arrow Function

ES6:

* JS
→ ecma script

Arrow function - under es 6.
* will not have function name

writing of arrow function:

```
<script>
const Reshowareyou = () => {
    return 100;
};
```

(Reshowareyou: JS variable / variable name)

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

document: object; getElementById: method; "response": Id name; innerHTML: Property; howareyou: variable; getElementById: HTML element; innerHTML: Way to display output.

* From ES6
- for effcing in terms of space and to increase the readability
- We can create functions without name and it is called as arrow function.

① Develop a calculator by getting 2 nos as input & display add, sub, product, quotient, remainder by creating individual arrow function for same.

② Create an array by taking array size and element from the user extract all the perfect numbers and even prime numbers from the array.

```
<html>
<script>
var arr = [1,2,3]
alert(arr)
arr.push(30)
arr.push(30,100,200)
alert(arr)
arr.pop
alert(arr)
```

Push → adding

Pop → delete last element

shift → delete first element

unshift → adding element in front

Inbuilt functions / methods

Splice

0 1 2 3

[1, 2, 3, 4]

9

(2, 2, 9)

(0, 1, 9)

Slice

n-1

[0, 1, 2, 3]

(0, 2)

Ans: (0, 1)

OOPS

Object Oriented Programming structure

Eg:- Birds → class

* Peacock
* cuckoo
* Sparrow

→ objects

flying
eating
wings
legs
Colour

→ Property

Behaviours → Methods

* Eating
* flying

JS objects :-

JS PROMISE

PROMISE → Javascript Object

Two states
- → Resolved (Success)
- → Rejected (failure)

CallBacks → Resolve / Reject

Note:
i) Involking a function
ii) Callbacks.

08/01/25

Andhra bday

PersonA → 5km
B → 8km
C → 10km
} Party → Andhras home.

1) write a promise called Andhra - BP

distance Andhra → A = 5000 mts
" → B = 2000 mts
" → C = 8000 mts

Outputs:
C reached
B reached
A reached

Promise inbuilt methods:

* when there is more than one Promise in order to review them we can use promise inbuilt methods according to the requirements.

methods:
1) Promise.all
2) Promise.any
3) Promise.allsettled
4) 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.

Promise.allSettled :-

Will display 1 among these three states :-

i) fulfilled
ii) Rejected
iii) Pending

Promise.race :-

* even with false.

Closure :-

React.js :-

* Library or framework of Java Script

Eg:

* Netflix
* Amazon

HTML :

* Youtube
* TCS

2 important folders :-

1. Public folder
2. Source

3 important files :

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

Note:
As of now dont touch index files.

Note:-
Initially do or write your code in app.js.

DOM

React follows VDOM:

* Once here unlike HTML once DOM gets created the changes are manipulation what we do gets completed and only that part will be rerended.

* where as in HTML everytime we make change entire DOM will be rerended.

Web application

* In web application created by ~~react~~ js each and everything is called as component.

Types of component:
(i) functional component
(ii) class "

JSX :
* writing HTML inside js

09/01/25

PROPS AND STATES

* Every component will have Props and States.

Props :-
* It wont change
Eg: Name (Tata's Birsleri)

States :-
* It changes (OR) we can change it.
Eg: Water level in Bisleri bottle.
Initial state - Full. updated state: half current state: Empty.

FLIPKart Website:-

1. Home page:
   Grocery Mobile Fashion
   Components

Mobiles:
Component name → mobiles
Props: name, version, price
State: discount

Passing props between components :-

* Create two files (i) Parent.js
  (ii) child.js

React Hooks

Earlier in IT industry used → class component
Reason being was state concept was not available with functional component.

→ Now Hooks
used to implement states in functional component.

types:
(i) UseState
(ii) Used Effect
(iii) Used Ref
(iv) UseContext
(v) Used Reducer

Eg: Used state
⇒ Counter clock
⇒ Stating the initial state as zero we can implement it, increment it, decrement it, reset it using used state hook.
→ takes one argument (initial state)
→ Returns ⇒ Array of two values Anything (empty) (initial & updated)

10/01/24

Spread operator:-

(i) Passing array in to use state

11/01/25

Use effect:

*upon the condition or action we apply in our functional components monitoring the impact of side effects can be done using use effect hook.

Accepts 2 arguments:

1. Callback function
2. Dependency array (optional)

Note:

1. Callback function is ~~known as~~ like constructor in JAVA.

20/1/2025

1. Create 5 Component: C1, C2, C3, C4, C5 and add + display messages

Component 1, 2, 3, 4, 5 respectively by keeping C1 as parent and rest all children as per the order. So C1 child as C2

C2 → C3, C3 → C4, C4 → C5 - Every component should display a message its name as message as C1 C2 C3 C4 C5. And display them side wise from h1 to h5

Note:

* when ever we are using something inside curly braces { }, it can be either JS ~~component~~ object or react component

* Props can be passed b/n components only by following the hierarchy.

which means parent → to child.

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

Conclusion:-

If we want to use state

The only way to achive is passing it as

props in the ~~hierarchy~~ hierarchy.

this is not efficient, to make it efficient we

had an exclusive hook called use context.

① 4 Components

1 ⟶ Component ⟶ App.js

2 ⟶ Container

3 ⟶ ~~app~~ Users

4 ⟶ User

Title, ③

without following the hierarchy passing

state from one component to another in an

efficient way using this Hook

Two important thing:-

① Create context

② Use context

In the Qn example: Create context will be

done in app component and ~~use to~~ that will be

used in User component using use context.

use effect:-

Synchronising ~~a~~ component with an

external system.

After one action monitoring or ~~st~~ seeing

the side effects happening in the functional

component is possible using use effect hook.

21/01/25

UseReducer :

- Same as useState to manage or update states that is data that is values of component.
- Difference is if you have more states or complex things you use usereducer hook.

Step1:
Create increment/decrement program using usestate.

Step2:
Replace usestate with use reduce.

Takes two argument:
* Reducer function (+, -)
* Initial value of the state (usestate)

State

Returns any array with two values
* initial state value
* dispatch function.

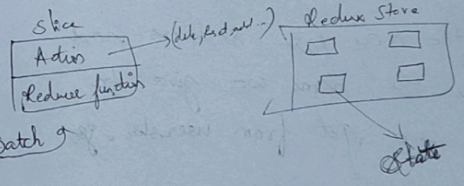
We call them as state & dispatch

The state will hold initial value and updated one you call dispatch function and dispatch will trigger use reducer function.

Get Pw from user, if pw is correct, display the component login granted. if pw incorrect display access denied component.

23/01/2025

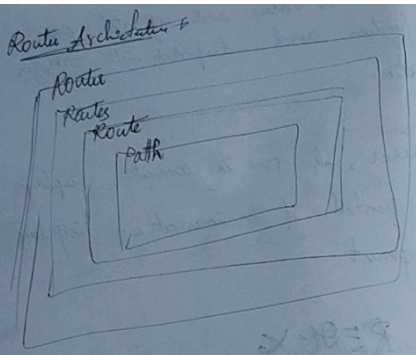
REDUX



To create Redux store → library is from js

(reduxjs/toolkit)

Route Architecture



Explanation:

Reducer: (reducers)

User_I&O is the key for reducers and userReducer is the name we give for reducer actions we get from userslice. js

After connection is done, use the hook called useSelector

State is reducer/userinfo from store/users from slice.

State is a callback function

useselector from store

useDispatch from slice

24/1

MongoDB

* json, we use NoSQL to process unstructured data

↓

js object

Eg: json

→ json file look like js object.

Compass: Help to fetch data from mongodb server, helps to reach to mongo db (helps to reach you to as its client).

Mongosh: interactive environment, Mongoshell is replaced with MONGOSH we can run queries do, perform administrative tasks

Data Modelling: → Nothing but fix structure of your data planning the structure

Eg: name, id, password.

Schema:

Actual blueprint of DB which you created by fixing the format using datamodelling.

Eg: Empdetails (name, id, salary)

SQL — Record

Mongo - Document

SQL - Table

Mongo - collection

use aiml

use employe → employe → emp

Show dbs

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

contact, salary → float

db.emp.find() → display all.

Commands: keywords

Create a db → db name → computers

Collection name → laptops.

↳ fields: name
model
Color
Status → Available / not available
Price
Vendor → Vname
Vprice

① List out particular model laptop

Change its status ~~to~~ unavailable → available.

25/01/25

db: Computer Collection: database

autoincrement ID, name, age, city

number

db.——.find({$and:[{Price:{$gt:80000}},{brand:'apple'}]})

db.——find({$or:[{Price:{$lt:10000}},{brand:'apple'}]})

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

——→ all documents in the collection where the hobbies field does not exist and the age field is greater than 40.

* Aggregation is group by
↳ in my SQL
↓
In MongoDB.

local field belongs to employees and foreign field companies.

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

→ new matched document will be added as an array in companies -- and the array name is employees.

① Create a db called bank
↳ 2 collections → ① Customer-personal
→ ② Customer-account

Model ①. name (string), address (array), phone (object) ↳ age, ② phone no ① & ②

② → Account no (int), branch (string), branch name (string), IFSC code (string), account balance (float), account type (string) ↳ Savings & Current, Overdraft (Y or N)

① Fill only OD → yes

② Display only the customers addresses where the branch names are same.

③ Current balance 10k → 20k filter only by phone numbers.

④ Filter only the Savings account peoples

⑤ Add a field called balance ($sum)
↳ collection
↓
only for IFSC code same people

Backend

Node → backend lib from js.

In node we can use, express js as midware.

Node

* Popular js lib
  → Or Run time environment → allows us to run js at server side.

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

Run command:-
* Start the server first
  Command: node server.js
  " to run client: npm start

Command in new terminal
→ npm i express

Const express = require('express');
↳ Keyword from js

"get" → requesting from server

"Post" → give info to server

app.listen(port, () => {
↳ keyword

Command:
we can delete package.json to get it back
→ npm init -y

Axios:

* Popular js library

→ used to make http request from the browser or node.js (client & server)

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

→ when we write an API for an exclusive purpose, we call it as rest API.

Cors:-

→ Cross origin resource sharing

• A web page requests information from resources (from any other sites) whether to accept the request and process it or not, will be defined in a rule book for this purpose we use Cors.

Commands:

npm i axios
npm i cors

① In the given example we are requesting data from server. hello this is 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 the message. So we are using from Cors. from response. msg.

Mongodb → app

Inside Src

Create 3 files:

- Users.js
- Createuser.js
- Updateusers.js

→ model, name, age, email

button → add users

Mongodb

Router, axios, express → cors.

N App.js bring

→ Routing for three components

→ Create Server folder

folder → inside that path

npm init → command.

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

Dependencies:

npm i react-router-dom

npm i axios

npm i express

npm i cors.

Create another collection inside db

db → empl

→ empid

empname

empsalary

Context → array (2)