JAVASCRIPTs

⇒ Js can be used at client side and service side

⇒ Js one of the powerfull language

⇒ React and Angular Js are popular frameworks or libraries of Js

→ Python is also a scripting language

⇒ <Script>
is used for include Js in HTML

```
<script>
    console. log ("  ")
</script>
```

✱ alert box

```
<script>
    alert(" box1 ")
    alert(" box2 ")
</script>
```

✱ Using datatypes

```
<scripts>
var num =100
console .log(num) // to view the number
alert( num)  // to view
</scripts>
```

* To print multiple values in alert

```
<html>
<body>
<h1> AIML Rocks </h1>
</body>
<script>
var a = 100;
let b = 200;
const c = 300;

alert ("a:" + a + ", b:" + b + ", c:" + c
alert (""+a+" " +b+" " +c)
console.log (a, b, c)
</scripte>
</html>
```

*
```
<script>
var a = 100;
a = a+ 100;
alert ( a )
</script>
```

*
```
<script>
let a = 100;
a = a+100;
alert (a);
</script>
```

*
```
<script>
const a = 100
a = a+100;
alert (a)
</script>
```
Error : values ca
be changed.

* Global scope   * local Scope

function

Eg:
```
<script>
var x =100;
function test () {
if (x == 100) {
var a = 10;
let b = 20;
console.log (a);
console.log(b);
}
console.log(a);
console.log(b);
}
test();
```
o/p: 10
10
10
b is not defined.

Q Do Two example prgms for each of the following

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

Q Case 1:

No. of Lemons in hand 7

Expected o/p:

God 1: 7 OFFERED
God 2: Need 7
God 3: Need 7

Shortage : 14

Case 2:

No. of Lemons in hand 21

Expected o/p:

God 1: 7 OFFERED
God 2: ~~Ne~~ 7 OFFERED
God 3: 7 OFFERED

Sufficient

Case 3:

No. of lemons in hand 15
God 1 : 7 offered
2 : "
3 : 1 offered & 6 needed [having 1 need 6]

Shortage : 6

Case 4:

No. of lemons in hand 67
God 1 : 7 offered
2 : "
3 : "

Surplus : 46

Q Sam is having 75 Candies. Sam gives half of it to his friend angle, since angle love sam a lot she gives back half of it. Calculate and display and individual and now much Candies they have.

Constrain :

Use one variable and one line calculation

ES6

* Extended version of javascript know as ECMA Script6.

Git folder

Step ① :- Day wise create a folder
[6 JAN 2025] → format

[Content] → upload notes

[handson] → programs, assignment

Console -> tracking the flow of program

Dom -> Document Object Model

Arrow function

⇒ * Es 6

* arrow function - under es6
  ↳ efficient ↳ function keyword will not be there.

id :
Uniquely Identify a single element.

* all the element in Java and python are called object

* In html called = element

```
<html>
<body>
    <h1 id = " response " ></h1>
<h1 id= "res"></h1>.
</body>

<script>
const howareyou = () => {
      return 100;        arrow function
};
var add = (a, b) => {return a+b}
document.getElementById ("response"). innerHTML =
 object          method              property
                                         howareyou ();
document.getElementById ("res"). inner_HTML =
                                         add (100, 200);

</script>
</html>
```

Arrow function

* Efficient space
* increase the readability
* Create function without name and it is called as Arrowfunction

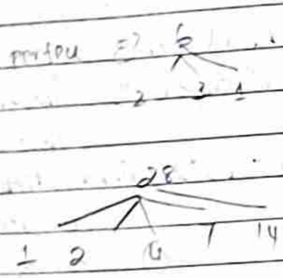
$\underbrace{\text{document}}_{\text{Object}}.\underbrace{\text{getElementById}}_{\text{method}}(\text{"response"}).\underbrace{\text{innerHTML}}_{\text{property}}=$ howareyou();

innerHTML

is used to insert (or) update the content inside an HTML element. Specifically it allow you to change the content dynamically using Javascript

Q Design a simple calculator by getting 2 numbers as input. Display addition, Subtraction, product, Quotent, reminder by Creating individual allow function of the same

Q Creak an array size, and by taking array size and element from the user extract all the perfect numbers and even prime number from the array



= 2+3+1 = 6

= 14+7+4+2+1 = 28

delete
* arr.shift( ) → add in the front of the array

* arr.unshift() => add element in the front of the array

⇒ Splice ()

```
Var Eis = [1,2,3,4,5,6]
alert (Eis)
Eis.splice (2,0,99)
alert (Eis)

o/p     1,2,99,3,4,5,6
Eis.splice (3,1,99)
alert (Eis)
o/p     1,2,99,99,4,5,6
```

```
Eis.splice (3,2,100)
alert (Eis)

o/p : 1,2,99,100,5,6
```

⇒ Slice

```
<html>
<script>
var arr =[1,2,3,4,5]
   var a = arr.slice (0,2) // print those number
alert (a)
alert (arr)
</script>
</html>

O/P : 1,2
```

⇒ Oops
Objects Oriented programming Structure

Ex :

Class ⇒ Bird
⇓
Object → { Peacock, Parrot, Sparrow }
⇓
Properties ⇒ { color, wings, legs }
⇓
Behavior (method) { flying, singing, Eating }

Java Script objects

⇒ Key and values

Object inbuilt method
1. Key
2. Values
3. Enteries

```
<html>
<Script>
```

Promise
⇒ ~~JS object~~
⇒ ~~2 State~~
~~Resolved~~ ~~Rejected~~

Js Promise

* Promise is a javascript object

* It has 2 States

Resolved [Success] Rejected [failure]

callback

* Invoking the func.
* (or) callback

Schedual - use priority queue.

```
<html>
< Script>
let thatkalbooking = new Promise ((resolve, reject) => {
let booking = false
if (booking)
resolve()
else
reject()
})
thatkalbooking then (success)
.catch (failure)
function success () {
console log (" thanks I send money") }
function failure() {
console log ("thanks for trying")
}
</script>
</html
```

8/1/25

Q Write a promise called andhara - Birthday Party (BP)
Function - peson - A, B, C
Distance } andhara - A = 5000 }
andhara - B = 2000 } Time Taken
andhara - c = 8000 }

Promise Inbuilt method

* Promises all(...)

When the is more than one promise in order to review them we can use promise in build method according to the requirments.

* promise.all()
* promise.any()
* promise.allsettled()
* promise.race()

Soln. Promise.all()

* If there are multiple promise if one is false the other will not be printed

=> One it see a promise false it will stop.

Promise.any()

* Gives the output ~~wi~~ if the where time is lesser will print first

=> Gives the shortest time promise provided status should True.

=> If all are false the gives an error
o/p: AggregateError: All promises were rejected

Promise.allSettled()

* all true

```
[ {..}, {..}, {..} ]
0: {Status: "fulfilled", value: 'A reached'}
                        "   : "B     "   }
1: {.                   "   : "C     "   }
2: {  "          "      ,
```

* If any is false

```
[              ]
0: { Status: "fulfilled", value; "A reached"}
1:      "     : "rejected", reason: "B not reached"}
                                                   }
2: {
```

will display 1 array 3 status
1) fullfilled
2) rejected
3) pending (can't seen in the output) in the mid state.

Promise.race()

* Same as Promise.any() but the difference is Promise.any() only work in true state

* But in race() if false it will print the false state.

* Closure

* function that remembers the variable from its lexical scope. even after the outer fun has finished executing. Closures allow inner function to access variables and parameters of their function, even after the outer function has returned

* React and HTML
=> React is slow compare to HTML
=> We are using react for the website which are dynamic changes more.

Defination: It is library or framework of javascript
Eg: Netflix and amazon.
HTML Eg: Youtube and wikipedia

React command.

① node -v

② npm (node package manger)
npm -v

③ npx Create-react-app demo
filename

④ npm start:
No opened so,
~~To open vs (command prompt)~~
To make it work

① Open ur react folder
u will see ur app folder got created

① Go to same above cmd prompt
② cd demo
③ code.

In vs

① Terminal
② New Terminal
③ npm start (error)
④ npm i web-vitals (error)

Go to commad prompt
① npm i web-vitals

Under VS code
① open:
=> Src
=> App.js (Change Learn React -> ATRL Rocks)

=> 2 important folder in react
* Public folder
* Src folder

2 component.

1. Class component
2 functional component

=> 3 important files.
* index.html
* index.Js
* index.css

=> As of now don't touch index file

Note:
Initially do or write ur code app.js

=> DOM
* react follow - V-DOM (virtual DOM)

* Here unlike HTML once dom gets created the changes (or) the manupilation what we do gets completed and only that part will be rendered

* Where as in HTML every time we make change entire dom will be rendered.

Web application

* Web application created by reactc Js each an every thing is called Component

Types of Component
- * functional Component
- * Class Component

XML
extensible Markup language
* Data transfer from one system to other

Props and State

Every component will have props and states

⇒ Props
* It won't change
Eg: name : TATA's Bisler

⇒ States
* It changes or we can change it
Eg: Waterbottel level in bottel
Initial state (or) full, updated state 1/2

Flipkart web Site

(1) Homepage → {Grocery, Mobiles, fashion} Components.
↓
(2) Mobiles
Component name Mobile
Propes {name, version, prise}
State {Discount}, {Stock}

Passing props b/w compots

React Hooks

Earlier in IT industry they were using class component, reason being state consept was not avaliable with functional components.

Now Hooks used to implement state in functional Component

Types:
* UseState
* UsedEffect
* UsedRef
* UseContext
* UseReduce

Eg for useState
Counter Clock
⇒ Stating the initial as 0 we can ~~implement~~ increment it, decrement it, reset it using used state Hook
* UseState takes only initialstate, only one argument
* returns: an array of 2 values
initial state and updated state

Spread operator

```
Const array1 = [1,2,3];
Const array2 = [4,5,6];

Const CombinedArray - [...array1, ... array2];

Console.log ("Array1:", array1);
            ("Array2:", array2);
            ("Combined Array:", combined Array
```

11/1/25

* use - Effect -1 :

Upon the condition (or) action we apply on function components monitering the impact or Side ffects can be done using useEffect hook

useEffect accept 2 argument
- → Callback function (is costructor in Java)
- → dependency Array

11/1/25
Q 5 Components names them as C1, C2, C3, C4, C5

npx create - react - app . <name>

C1 → childis C2 → child is C3 → C4 → C5

every component should display its name as mesg.

* if u can alse use direct export
Eg: export const C1 = () => {

* Whenever we are using something inside the { } it can be either js component (or) react component

* Props can be passed b/w components only by following the hierecy, which means parent to child

* To over come this in the team of efficiency we are using hooks

conclusion:
if we want to use

* The only way to achive it is passing it in hiercey.

* This is not efficient, to make it efficient exclusive hooks called useContent

4 Component

Component 1 = App.js
Component2 = Container
Component3 = Users
Component 4 = User

To use UseState

Step① - import useState in App.js

Step② - Same in App.js after func. App()
const [theme, setTheme] = useState
("light")

UseContext

Without following the hierachy passing the state from the one component to another component in a effecent way using hooks

1. Create Content
2 UseContent

In the give eg CreateContent will be done in app component and that will used in userComponent using use Content

UseEffect

Synchronizing component with external system after our action monitoring or seeing the sideffect happing in the functional component is possible using useEffect hook

=> callback func
=> dependency array (optional

UseReducesur

Same as usestate to manage (or) update state that is data that is value of componen

difference is if u have more states or comple thing you use usereducer hook

Step1 :

Create increment decrement program using usestate

Step 2:

replace usestate with usereducer

Note

1. UseReducer takes 2 argument,
   ⇒ first is reducer function which say what you want to do (like increment (or) decrement),

   ⇒ Second is initial value value of state

2. usereducer returns array with two values like usestate.
   2 values
   ⇒ initial count and second is dispatch function.
   We call them as State and dispatch
   ⇒

   ⇒ Now state will hold initial Value and updated Once you call dispatch function and dispatch will triger UseReducer function

Q. Display 2 variable value on Screen
↳ is object x & y

Q Get a password form user as input if the password is correct display the component login granted

if password incorrect display access denied

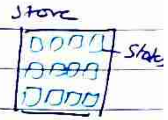
Redux

* frame work

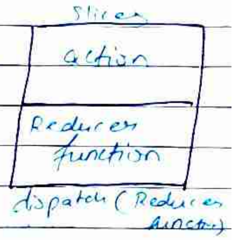
⇒ managing the state of the react component

⇒ Global · start managable system

⇒ Redux also manitain store containing states



⇒ Redux also contain Slices



Intallation for Redux.

npm i @reduxjs/toolkit react-redux

⇒ to create redux store and all then given one library than is reduxjs/toolkit-

* event. target. value
JS object.

Bootstrap.

npm install bootstrap react-bootstrap

Button creation

```
import {Button }. from 'react-bootstrap'
        (or)
import Button from 'react-bootstrap/Button';
```

```
<Breadcrumb. Item active> Home </Breadcrumb
                                 Item>
```

if we give active
the link will get deactive

Container card.

NoSqL

* Unstructured data

Eg : JSON & file

JSON file will look like js object
Javascript Object Notation.

=> Compass

Compass helps to fetch data from mongodb server, means compass helps u to reach mongo db like its clint

=> Mongosh

* Mongo shell was replaced by Mongosh

* The ~~MongoDB~~ Mongosh ~~Shell~~ gives you an interactive environment where you can run que

=> 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 datamodlling

Eg: Empdetails (name, id, salary)

{Ctrl + Shift + Q

⇒ Open Mongosh

Commands ~~t~~ :
⇒ Use `aiml

⇒ Go to Mongo DB

⇒ create new file <sahana>
⇒ Near sahana '+' click on it
* add db-name as ". employee"
* add db collection name as " emp"

⇒ Go to Mongosh

⇒ Use employee
⇒ vs Show dbs

* Add data

⇒ db. emp. insertOne ({ empname : " Sam " })

⇒ db. emp. find ()
to print all the element in Cmd

⇒ db. emp. find(). pretty()
to see output in Structured way

⇒ Cls
clear the screen

classmate
Date
Page

embadded document
⇒ db. emp insertOne ({ name : " Sam ", contact: 789 , lik : " Sport ", favs : { game : "football", player : "Ronaldo" }})
⇒ db emp. updateOne ({ empname: " }, {$set:{}})

New file [10 records]

* Create db → Computers
* Collection → Laptops
fields
Name
model
color
Status (available , Not available)
price

vender → vendar name
vendarprice

Q. list out particular model lapatop
Q. Change its status to available → Not availa ble,

add as array

⇒ db. laptop. insertOne ({ Variant : [" v10,"
" v11 ", " v12 "]}

Replace
db mycollection - replace One (
{ name : "John" } { name : " John ",
age : 32, city : " New city " }

find

```
db.Laptop.find({ name: "Dell "})
```

⇒ give all the dell named laps

FindOne

```
db.laptop.findOne({ name : "Dell "
```

⇒ give 1st record named as. dell

Hi Sahanaa!!! :)

```
⇒ db. customers. find ({hobbies : { $nin : [ 'cooking']
```

```
⇒ db. products. find ({ $ and : [{ price : { $ gt :
  }, {brand : "Apple" }]})
```

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

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

5|1|25

DB Computeras Collect: Details

```
"_id" : 3,
"name" :
"age" :
"city" :
```

* aggregation ⇒ is group by

```
⇒ db. Companies . aggregate ([{ $ lookup : { from: "Emply
  , localField : 'id', foreignField : 'company_id', as
  : "Employee" }}])
```

Create a database Bank
⇒ 2 collection under that
Names of collection 1. Customer personal
2. Personal Customer acc

1. Data Mode for custome personal
_id
Name : string
address : array
Ph. No : Object (Name them as 1 and 2
Age :

_id
Acc No : int
Branch : String
Bank Name : String
IFCE : String (
Curr Balance : float
Acc type : Saving and current
OD (over draff) : Yes (or) No

1. Od : Yes
2. Display only the customer adder
   where branch Names are same
3. Current balanc 10000 to 20000 (print phNo.]
4. Filter only the Saving acc people
5. Add filed called status (value : Same) add
to those IFCE code : Same

NODE.JS

* Backend is also a middleware

* Node is backend lib from js in node we can use express js as middleware

TO start NODE.js

⇒ node <file name>

```
st http = require ('http'),  // Built-in module
 express = require ('express'). // Third-party module
sayHello = require ('./greet') // custom module
```

27/1/25

Request : from client
Respond : from server

1st start the server (npm start)
2ed start the client (node<filename)

* in server we don't have autosave so v use Nodemon

⇒ its a runtime env the allow you to run JS

NOTE:
Maintain split teminals in vscode in order to use client and server

Run Comment
always start the server first
node server.js

for node → npm start

To start

① npm i express

* Sower also gives json file

* To activate tee port app.listen
file name

* we can delet package.json to get it back
~~Node~~ npm init -y

⇒ Req is from ~~Nodejs~~ react, res from Node js
⇒ Create an instance of ~~my~~ an Express
application

⇒ Used for routing, middleware managemen)
and may reasons

⇒ to keep routing clean

Const app = express ();

⇒ // Define the port number the server
will listen on
Const port = 3000;

⇒ // Define a route for the HTTP GET request
to the rout URL
('/').

⇒ // req represents the ~~pro~~ request object, and
res represent the response object

⇒ means browser asking the server (
so it the re

* It is popular js library used to make
http requires from the brower / client

* is known for easy & clean syntax
and also flexibility Especially works will
with API and Rest API

⇒ When we write API for exclusive purpose
we call it as "rest API"

CORS
⇒ Cross origine resources sharing

npm i -g nudemon
↳ nodemon server.js

npm i axios
npm i express cors

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

* In the gn example, we are requesting data from Server
"Hello, this is data from server.

Server respons has json from the json i want to filter only msg, so we are using cors respons.data.message

Axios

* It is popular Js Lib
* used to make HTTP request from the browers (or) node.js

Axios is known for easy and clear Syntax and also flexibility, especially works well with APIs and rest API.

Inside src create 3 file

1) users.js
2) Createuser.js
3) updateuser.js

Model
⇒ Name
⇒ Age
⇒ Email

Router
axios
Exprees } dependancy
Exprees cors

App.js → Add routing for 3 models

⇒ Create a folder called server.js inside the terminal
npm init

⇒ inside serve folder index.js backend code we have to write

⇒ npm i mongoose in vs terminal