FEBRUARY FULL STACK DEVELOPMENT TRAINING

----------01/02/2025 Saturday

1)Variables:

Let, var, const ;

2)Operations

3)Conditional Statements

4)loops

5)functions

6)OOPS,DataStructures

7)Ap,V,C

8)promises

JavaScript Revision Practice questions:

Scenario 1 :you are tasked with building a system to check whether a person is eligible to vote.The system follows these rules:

1.A person must be at least 18 years old to vote.

2.If the person is a citizen ,they are eligible to vote.

3.If the person is not a citizen but has passed a citizenship test,they are eligible to vote.

4.If the person is under 18,they cannot vote.

Code:

```
let age = prompt("Enter the age");
    let citizen = prompt("Enter the citizenship");
    let cTest = prompt("Citizenship result");
    if (age >= 18 && (citizen == "Indian" || cTest == "Passed")) {
      document.write("Can eligible for voting");
    } else {
      document.write("Cannot vote")
    }
```

Scenario 2:Discount Eligibility

An online store offers a discount system based on the following consitions:

1.A customer who spends more than Rs.100 gets a 20% discount.

2.A customer who spends more than Rs.50 but less than or equal to Rs.100 gets a 10% discount.

3.If the customer is a premium member, they receive an extra 5% discount.

Code:

```
let customer1 = parseFloat(
  prompt("Enter the total amount of bill for customer 1")
);
let customer2 = parseFloat(
  prompt("Enter the total amount of bill for customer 2")
);
let customer3 = prompt( "Are you a premium member? (yes/no)" ).toLowerCase();
if (customer1 > 100 ||(customer2 > 50 && customer2 <= 100) ||customer3 === "yes") {
  document.write("You get a 20% discount");
} else {
  document.write("You get a 10% discount");
}
```

----------**05/02/2025 Tuesday**

To create a element by React js:

---React.createElement('h1');

Syntax:

React.createElement('h1')

-type = tag name (div,h1,p)

Props = className,id,onClock

--to display the element

--ReactDOM.createRoot()

--ReactDOM.render() --to display the element

Syntax:

Reactelement -what to render

Container = where to render

Intro to JSX:

(JavaScript XML) is a syntax extension for js in react js;it allows us to write html code in react.

--make us easier to write Html in react.

JSX:JSX code gets compiled into JS

Babel:a tool converts translates into js

Keypoints;

Using function call

----------05/02/2025 wednesday

--ReactDOM.createRoot();

--render

JSX:

Javascript XML is a syntax extension for JS:

It allows us to write HTML code in React

The JSX code gets compiled into JS

Babel:A tool converts HTML code in JS

--All the html tags must be closed

----------06-02-2025 Thursday

#empty folder

#npx-nope package executor

create react app

-npx create-react-my-app #here my-appp is our project name

cd my-app #change directory(folder)

npmm start #to startthe react application

Public/index.html:

The main html file that serves as the entry point for the app.

Src/index.js: The js entry point for the react app where the DOM is rendered

Src/App.js: The main component that serves as the root of the component tree

Src/components: A folder to store reusable components

1)components core building blocks of a react application)

->The help to create reusable block of code

->If an thing goes wrong in UI, it is very easy

----------07/02/2025 Friday

Components:

Conditional Rendering:

If , else:

App.js:

```
import React from "react";

import Greeting from "./greeting";

function App() {
  const isLoggedIn = true;
  return (
    <div className="App">
      <Greeting isLoggedIn={isLoggedIn} />
    </div>
  );
}

export default App;
```

greeting.js:

```
import React from "react";

function greeting({ isLoggedIn }) {
  if (isLoggedIn) {
    return <h1>Welcome Back!</h1>;
  } else {
    return <h1>Please Login</h1>;
  }
}
export default greeting;
```

#error-scripts disabled-command;set

#web vitals --- npm install web-vitals

#lists and keys:

In Reat,a list is a collection of items you want to show on screen

#keys: Keys in react helps us to keep the track of items in a list

->React know which items you have changed; updated

Map(): elements in array

--function will be applied to all elements

Const n=[1,2,3,4,5]

Const d = n.map(num =>num*2)

Console.log(d)

#2,4,6,8,10

#map() in react for lists:

App.js:

```
import React from "react";

const FruitList = () => {
  const fruits = ["apple", "cherry", "orange"];
  return (
    <div>
      <h1>Fruits List</h1>
      <ul>
        {fruits.map((fruit, index) => (
          <li key={index}>{fruit}</li>
        ))}
      </ul>
    </div>
  );
};
export default FruitList;
```

----------**08/02/2025 Saturday**

State:

State in React:

In react, state is alike a container that holds the data or information for a component. This data can be change over time based on user actions or events.

Why state is important:

--It allows us the component to remember things

Ex: if you click a button to change a color; the state will store hold the color and show on the screen

1)functional components:

Syntax:

Const[statevariable ,setstatefunction]=useState(initialValue)

1)statevariable : holds the current state(ex: name,color)

2)setStatefunctions: A function which is used to update the state

3)initial function:The initial value of the state variable when the component first renders

----------**10/02/2025 Monday**

--to manage state and life cycle features in the functional components

State: is ana essential part in react because it allows components to be dynamic ,interactive and capable of responding to user input or change over time.

1)useStateHook:

useState allows you to add state to functional components.

Syntax:

Const[state ,setState]=useState(initial value)

State: This is current state

setState-This is a function to update the state

initial value : The value you want you set as the initial value

```jsx
import React, { useState } from "react";

const ThemeToggler = () => {
  const [theme, setTheme] = useState("light");

  const toggleTheme = () => {
    setTheme((prevTheme) => (prevTheme == "light" ? "dark" : "light"));
  };
  return (
    <div
      style={{
        backgroundColor: theme == "light" ? "white" : "black",
        color: theme == "light" ? "black" : "white",
        textAlign: "center",
      }}
    >
      <h1>The current Theme is{theme}</h1>
      <button onClick={toggleTheme}>toggleTheme</button>
    </div>
  );
};
export default ThemeToggler;

// import React ,{useState} from "react";
```

```jsx
import React ,{useState} from "react";

//countercomponent

const Counter = () => {
  const [count, setCount] = useState(0);
  //current state = count
  //functio  to update state = SetCount
  //use state() , ti initialize the state =0;

  return (
    <div>
      <h1>{count}</h1>
      <button onClick={() => setCount(count + 1)}>Increment</button>
      <button onClick={() => setCount(count - 1)}>Decrement</button>
    </div>
  );
};
export default Counter;
```

React Memorization:

----------12/02/2025 Wednesday

React memo:

--it is a HOC(high order component) is not a Reacr hook .

--it will stop unnecessary rendering of functional components of its props

--it will improve the performance of the functional components

#keyword

#suggestions

#1000 product

#render

App.js

C1

C2-increment

App.js:

```jsx
import childA from "./childA";
import childB from "./childB";

const Parent = () => {
  const [count, setCount] = useState(0);
  const increment = () => {
    setCount((c) => c + 1);
  };
  return (
    <div>
      <childA />
      <childB count={count} increment={increment} />
    </div>
  );
};
export default Parent;
```

childA.js:

```
const childA = () => {
  console.log("child A");
  return <h1>This is child A</h1>;
};
export default childA;
```

childB.js:

```
const childB = ({ count, increment }) => {
  console.log("child B rendered");
  return (
    <div>
      <h1>Count:{count}</h1>
      <hbutton onClick={increment}>Increment</hbutton>
    </div>
  );
};
export default childB;
```