PHASE 4 TOPICS

**Front End Basic TOPICS**

1. HTML => Understand the various tags in HTML
2. CSS => to style the web pages.
3. Bootstrap => a css library used for styling
4. Javascript
   1. Dialog boxes
      1. Alert
      2. Prompt
      3. Comfirm
   2. Variable : vat , let and const
   3. Operators
   4. == and ===
   5. Conditional constructs : if-else and switch-case
   6. Loops : for , while and do-while
   7. Arrays :
   8. Array methods
   9. String methods
   10. Functions
       1. Named
       2. Function expressions
       3. IIIFE
       4. Arrow
   11. ES6 Syntax
       1. Spread
       2. Rest
       3. Destructuring syntax
       4. Default arguments in functions
       5. Immutable and Mutable state of javascript properties
       6. Class
       7. Promise and AJAX
       8. Fetch for REST API

**REACT TOPICS**

1. Download VSCode
2. Download NodeJS
3. Using create-react-app, create a react project  
   npx create-react-app reactapp
4. Components in React : Reusable JSX functions or class that consists of View and BL
   1. Functional Based
   2. Class Based
5. Props : They are read only and passed as parameters to the components
   1. Class Based : this.props
   2. Functional : props (as they are passed as parameters)
6. State : They maintain the state of component within the class or function. They are like instance variables local to the component
   1. Class : this.state and define in the constructor   
      To modify the state value use setState()
   2. Functional : useState  
      To modify use the setter
7. Conditional Rendering :
   1. If
   2. Short circuit operator &&
8. Lifecylce of the component
   1. Class Based
      1. ComponentDidMount: called once at the start of the lifecycle
      2. shouldComponentUpdate : called anytime the props or state value changes and only if it returns true will the UI be updated
      3. componentDidUpdate : if shouldComponentUpdate returns true then this method is called and component is updated
      4. componentWillUnMount : Called once the lifecycle when component is removed from the DOM hierarchy
   2. Functional Based:
      1. useEffect(()=>{}) : With only 1 parameter is same as all lifecycle methods in class based and will be called again and again
      2. useEffect(()=>{}) : With 2 parameters and 2nd parameter as empty array is same as componentDidMount and will be called only once
      3. useEffect(()=>{}) : With 2 parameter 2 and 2nd parameter with a prop or state name in array is same as componentdidmouint and for the changes in array variable is same as shouldComponentUpdate and componentDidUpdate
      4. useEffect(()=>{}) => If first parameter returns a function then that is componenetWillUnmount
9. Form Handling:
   1. React controlled form components by associating the value attribute with state of the component and handle onChange event for updating the state
   2. React uncontrolled form components by not associating the value attribute with state of the component and not to handle onChange event for updating the state
   3. Onsubmit event: gets fired whenever the submit button is clicked
   4. Event of the element that is clicked is an object that is automaticalluy passed to the event handlers
   5. For form submission, we used event.preventDefault() to basically prevent the default behavior of the browser
10. List handling:
    1. Iterate over the list using JSX syntax and the map function of the arrays
    2. To provide the key attribute for every html element in the list that represents every item in the list. This allows react to handle the updates efficiently
11. State of a react application is always kept at the root component (App) and the state flows from App to its child component via props as data and from child to parent as event handlers
12. Routing
    1. Npm I react-router-dom
    2. Import BrowserRouter , Routes , Route
    3. Wrap all the components and Routes within the BrowserRouter to take advantages of routing functionality . hence we added BrowserRouter in App
    4. Using Routes create Route, provide path and the component to be displayed for the respective path
    5. Using Link or NavLink can create the links
    6. To pass parameters from 1 route to another, create the url with the placeholder like : <Route path=’/:id’ element={}/>
    7. To access the path parameters , react router provides with useParams as a hook to access the value from the parameters
    8. To access query params i.e with ? it provides with createSeacrhParams and useSearchParams
13. Redux
    1. Provides with a store for the state of the application
    2. Reducers are pure functions that are connected with the store and modify the state of the store immutably
    3. Requests are dispatched to the reducers with the action to be performed i.e CRUD along with the payload
    4. Steps for learning redux
       1. Folder => reduxdemo
       2. From within reduxdemo initialized node project  
          npm init -y
       3. Installed redux  
          npm i redux
14. Redux with react
    1. Within the react project install  
       npm i react-redux  
       npm i @reduxjs/toolkit
    2. Reducers folder within which we have the state and the reducer function using createSlice
    3. Store.js file where the reducer was connected with the store
    4. In index.js, to make the store available to react root component using Provider
    5. To connect the store with react component
       1. Class based : create 2 functions to map the state and the dispatch functions / actions with the UI
          1. mapStateToprops => this.state.props.count
          2. mapdispatchtoprops => this.state.props.increment()
       2. functional based :
          1. useSelector() : that returns the state as props : props.count
          2. useDispacth() : dispatch(increment())