LIBRARY : COLLECTION OF PREDEFINED CODES

FRAMEWORK : COLLECTION OF LIBRARIES

*****REACT JS IS A JAVASCRIPT LIBRARY******

ReactJs is used to make single-page applications.

React has a component based architecture.

Facebook (Jordan Walke) introduced React in the year 2011 and made it as an open source in 2013.

Single Page App:

1. Faster Loading Time
2. Fast Development
3. Fast and easy maintenance
4. Limited SEO(Search Engine Optimization)
5. More Expensive

Multi Page App:

1. Slower Loading Time
2. Slow Development
3. Slower maintenance
4. Easier & More effective
5. Less Expensive

Features of react:

1] Declarative

2] Component Based Architecture

3] Unidirectional data flow

4] Learn once, write anywhere

5] Virtual DOM

Virtual DOM

1. Virtual DOM is a copy of real DOM, you can say a xerox copy.
2. Whatever updation we do on the virtual dom is not reflected on the real dom or the screen directly.
3. It does all the manipulation within itself and then render those changes on the real dom so in this way we do not need to update the real dom again and again.
4. It can change 2,00,000 nodes/sec

Diffing :

React compares the virtual copy of the real dom to an updated copy of virtual dom , compares or picks out the changes. This process is called as diffing and the algorithm used is called as diffing algorithm.

Reconciliation:

When a component(node) changes , react decides whether it should render the changes on real dom or not.

So if the states/props of two nodes/components are not same, then it renders the changes to Real DOM.

This process is called as Reconciliation.

In react, when the state of a component changes, the component needs to update its UI to reflect new state.

This process of updating the UI is called as reconciliation.

It is dependent on :

1. Virtual DOM
2. Diffing Algorithm

Installation of vite

1]Install node JS (after installing close everything and install it only once)

2]Create a folder

3]Open with cmd(Command Prompt)

4] We need to pass a command : npm create vite@latest

5]Ok to proceed? y?

6] project name: project-name

7] Select the framework: React

8] Select variant:JavaScript

9] cd project-name

10] npm install

11]open with vs code

12] Run the code : npm run dev

Why React JS?

1. To create Single page applications.
2. It is Declarative
3. It follows Component Based Architecture.

Folder Structure:

node-modules: All the pre-defined codes are present in this folder(do not touch it)

src :

1] It is a source folder where we are going to write the code.

2] Inside src you have two important files i.e

i. main.jsx :This will create a connection between the src and index.html

ii. App.jsx: It is a component where we will be writing our code.

package-lock.json & package.json: These are considered as directories of the react folder.It will provide you information about libraries present in the project.

JSX: JavaScript Extensible Markup Language

It is a syntax extension for JavaScript.

It is a template language of React.

Rules of JSX:

1. JSX must be closed, which means each and every element should be closed.

   <h1>.....</h1>

   Even if it is a void tag or empty tag then close the element there itself.

   <input/>

2. JSX must have one parent element or else you can get one error.

   Adjacent elements should be wrapped in an enclosing tag.

```
<div>
<h1>JSX....</h1>
<h2>JSX....</h2>
</div>
```

3. The tagnames should always be in lowercase.

   <h1>...</h1>

   <H1>....</H1> Throws an error

4. In html we used attribute 'class' and 'for'. But in jsx we will use 'className' and 'htmlFor'.

5. All the events we use should be in lowerCamel case.

   Ex: onClick, onSubmit, onMouseOver

JSX Expressions:

1. With JSX you can write expressions inside curly braces {}.
2. The expression can be a react variable, or property or any other valid javaScript expression.
3. JSX will execute the expression and return the result.

COMPONENTS:

1.Components are nothing but building blocks of the web page.

2.Webpage will be divided into multiple components(files) and then we will be joining together in the parent component (App.jsx).

3. Components are reusable.

Rules while creating a component:

1. First letter of the file name should always be capital.
2. File extension of component is .jsx
3. We can declare components in two ways:

   i. Self-closing tag (<ComponentName/>)

ii. Paired tag (<ComponentName></ComponentName>)

Types of Components:

1. Functional Based Component
2. Class Based Component

Functional Based Component:

1. It uses javaScript function.
2. FBC is stateless.
3. We can use Hooks.
4. Can't use life-cycle methods.
5. render() is not used.
6. const Func = ()=>{

```
return (
<h1>Hello</h1>
      )
    }
```

Class Based Component:

1. It uses class.
2. CBC is statefull.
3. We cannot use Hooks.
4. We use life-cycle methods.
5. render() is used.
6. import {Component} from 'react'

```
  class CcomponentName extends Component {
   render(){
        return(
            <h1> hello </h1> )
          }
          }
```

PROPS

1. Props are nothing but properties.

2. Props are used for transfering data from parent component to child component.

3. Props follow unidirectional dataflow i.e from parent to child component.

4. It is an inbuilt object.

5. Props are immutable , it means once the data is passed from parent component it can't be changed.

FRAGMENT

1. Fragment let you group a list of children without adding extra node to the DOM.

2. Fragment can accept only "key" prop as a property.

3. Except this it will not accept any other properties.

```
1] import React from 'react'
      <React.Fragment>
       <h1>You can give only one prop i.e key</h1>
       <h2>Functional Based Component</h2>
     </React.Fragment>
```

```
2] import {Fragment} from 'react'
      <Fragment>
       <h1>You can give only one prop i.e key</h1>
       <h2>Functional Based Component</h2>
      </Fragment>
```

```
3]
     <>
       <h1>You cannot give any props</h1>
       <h2>Functional Based Component</h2>
     </>
```

** key prop provides unique identity for each data(value) inside the array which you are mapping.

** PROPS DRILLING **

1.Prop drilling occurs when a parent component passes data down to its children and then those children pass data down to their own children.

2.At the end its a long chain of component dependencies that can be difficult to manage and maintain.

3. The problem with this approach is that most of the components through which data is passed have no actual requirement.

4. They are simply used as mediums to transfer the data to the destination component.

5. We have an alternative for prop drilling i.e Context API.

** {props.children} **

We can use props.children on components that represent 'generic boxes' and that do not know their children ahead of time. It is used to display whatever you include between opening and closing tag when you render the component.

**Default Props: The defaultProps is a react property that allows you to set default values for the props argument.

Proptypes*

Proptypes are a mechanism to ensure that components use the correct datatype and pass the right type of data and the component uses the right type of props and that receiving components receive the right type of props.

Command to install : npm install prop-types --save

import Proptype from 'prop-types'

STATES

1. State is a javascript object used by react to represent the information about the component's current situation.

2. States are used to create dynamic data on UI.

3. States are mutable, states values can be changed.

4. States are local , states belong to one particular component.

5. By default, FBC are stateless.

It means we dont have any inbuilt state object in FBC.

But we can make it statefull , using an advanced hook from react library .

SYNTAX OF useState()

```
   let [state, setState] = useState("Hello")
```

state==> Variable Name

setState==> It is a function used to change the current value of "state" .

useState() ==> It is a hook

Hello ==> current value of state

Hooks

1.Hooks are like inbuilt methods in react.

2. Hooks always start from a prefix word "use".

3. Whenever we want to use hooks, we have to import it from React library, import statement is mandatory.

4.Hooks are only used in Functional Based Components.

useState() hook is used to create state in FBC.

******** props & states******

PROPS:

1.It passes data from one component to another component.

2. Readability : Read only

3. Mutability : Immutable

4. Child component can access.

STATES:

1. It holds information about the component.

2. Readability : Changeable

3. Mutability : Mutable

4. Child component cannot access.

CSS IN REACT

1. INLINE CSS

It is a type of css which will apply individually inside one particular tag using "style" attribute.

The CSS properties should be written inside an expression in the form of 'object'.

2. GLOBAL CSS

It is a type of css which we will use to maintain one css file for entire react project .

It will target all the components.

We have to create a separate css file inside 'src' with an extension of '.css' and write all the styles.

3. MODULE CSS

We will be creating a separate css file for each component.

The respective styles required for the particular component will be written in their respective css file.

Whenever we are using module css we have to create css file with an extension '.module.css'

For eg. filename.module.css

CONDITIONAL RENDERING

In react, conditional rendering is the process of displaying different content based on certain conditions or states. It allows you to create dynamic user interface.

if-else , else-if , switch , ternary operator, short circuiting.

***ContextApi ***

1. Context is a way to share data between components without having to pass props down to the component tree.

2. useContext is a hook that allows you to consume context values in your functional components.

3. To create a context, use the createContext() method.This returns a context object that can be used to provide and consume values.

4. To provide a value to a child component, use the Context.Provider component. This component accepts a value prop that is passed down to child components.

5.To consume a value in a child component, use the useContext hook. This hook takes a context object as its argument and returns the current value of the context.

HOC(Higher Order Components)

HOCs are the functions that accept a component as an argument and return a new component with additional functionality.

With HOC pattern , we can easily create reusable components in Reactjs.

They allow you to reuse component logic across multiple components.

References

1. It is an inbuilt object in Reactjs.

2. It is used to target element in Reactjs.

3. Bydefault, it will be having key-value pairs of "current:undefined".

4. Ref will always use Real DOM.

5. In FBC , we use 'useRef' hook to create references.

6. We use refs in managing focus, text selection or media playback, etc

*** FORMS IN REACT****

There are two types of forms in React.

1. Uncontrolled Form

2. Controlled Form

1]Uncontrolled Forms:

1. It is a form where it is created using reference concept.

2. In FBC, we use 'useRef()' to create uncontrolled forms.

3. These forms are completely handled by DOM itself.

4. Suppose if we want to take any data from user, first the data will be taken by DOM directly and then we will be taking the data from DOM.

2]Controlled Forms:

1. In React, a controlled component is a component where form elements derive their value from a React state.

2. When a component is controlled, the value of form elements is stored in a state and any changes made to the value are immediately reflected in the state.

3. To create a controlled component, you need to use the value prop to set the value of form elements and the onChange event to handle changes made to the value.

4. The value prop sets the initial value of a form element, while the onChange event is triggered whenever the value of a form element changes.

5. Inside the onChange event, you need to update the state with the new value using a state update function.

STEPS:

1. Initialize state object.

2. Add name and value attribute to the input or form elements.

3. State Update (Add onChange event)

4. Add onSubmit event.

SYNTHETIC EVENT (e)

1.Synthetic events in React are cross-browser wrappers around the browser's original event or native events.

2.Different browsers may have different names for the event .

3.They create a uniform interface that React uses to ensure that events execute consistently across all the browsers.

4.React normalizes this information to provide a consistent API for handling events regardless of the browser.

LIFE CYCLE METHODS

1.In React, components have a lifecycle that consist of different phases.

2.Each phase has a set of lifecycle methods that are called at specific points in the component's lifecycle.

3. These methods allow you to control component's behavior and perform specific action at different stages of its lifecycle.

1] MOUNTING PHASE :

Mounting phase is a phase where an instance of a component is being created and inserted into the DOM.

i. constructor(props)

ii. static getDerivedStateFromProps(prop,state)

iii. render()

iv. componentDidMount()

2] UPDATING PHASE:

Updating phase is a phase when a component is being rendered as a result of changes either to its props or state.

i.static getDerivedStateFromProps(prop,state)

ii.shouldComponentUpdate()

iii. getSnapshotBeforeUpdate(prevProps, prevState)

iv. render()

v. componentDidUpdate(prevProps, prevState, snapshot)

3] UNMOUNTING PHASE :

Unmounting phase is a phase when a component is being removed from the DOM.

ERROR HANDLING :

Error handling is a phase when there is an error during rendering in a lifecycle method .

1] MOUNTING PHASE :

i. constructor(props)

* It is a special function that will get called whenever a new component is created.

* Best place to initialize state and bind the event handlers like 'this' keyword.

* Do not cause side effects. Ex.HTTP requests

ii.static getDerivedStateFromProps(prop,state)

* This method gets invoked when the state of the component depends on changes in props over time.

* Rarely used method.

* Do not cause side effects. Ex.HTTP requests

iii. render()

* This is the only required method.

* It will read props & state and return JSX

* Do not change state or interact with DOM or make ajax calls.

iv.componentDidMount()

* Invoked immediately after a component and all its children components have been rendered to the DOM.

* Can be used to cause side effects. Ex Interact with the DOM or perform any ajax calls to load data.

* It is rendered only once in the DOM.

2] UPDATING PHASE :

i. static getDerivedStateFromProps(prop,state)

* This method is called everytime a component is re-rendered.

* It sets the state.

* Do not cause side effects. Ex. HTTP requests

ii. shouldComponentUpdate(nextProps, nextState)

* This method dictates if the component should render or not.

* Do not cause side effects. Ex. HTTP request.

iii.getSnapshotBeforeUpdate(prevProps,prevState)

* This method is called right before the changes from the virtual DOM are to be reflected in DOM.

* It captures some information from the DOM.

* This method will either return null or return a value.

* The returned value will be passed as the third parameter to the componentDidUpdate method.

iv. render()

* Read states and props and return jsx.

v. componentDidUpdate(prevProps,prevState,snapshot)

* This method is called after the render in the re-render cycles.

* This method allows you to use side effects or can be used to cause side-effects.

3] UNMOUNTING PHASE :

i.componentWillUnmount() :

* This method is invoked immediately before a component is unmounted and destroyed.

* For example : Cancelling any network requests, removing event handlers, cancelling any subscription.

ERROR HANDLING PHASE:

When there is an error either during rendering in a lifecycle method or in the constructor of any child component this phase occurs.

i. static getDerivedStateFromError(error)

ii. componentDidCatch(error,info)

USEEFFECT HOOK

1. The useEffect hook is used to handle the side effects such as fetching data and updating DOM.

2. This hook runs on every render but there is also a way of using a dependency array using which you can control the effect of rendering.

3. You can call useEffect with a callback function that contains the side-effect logic.

4. By default, this function runs after every render of the component.

5. You can optionally provide a dependency array as the second argument.

6. The effect will only run again if any of the values in the dependency array change.

SYNTAX OF USE-EFFECT:

useEffect(callback,dependency)

```
1. useEffect(()=>{
            // You can write your code for mounting
      } , [] )
```

// Adding empty array ensures that the useEffect gets invoked only once (When component mounts)

```
2. useEffect(()=>{
                // You can write your code for mounting
        } , [values] )
```

// Adding dependency values invokes or triggers useEffect whenever they are updated in the program

// You can also give multiple values for dependencies by separating with comma.

SIDE-EFFECTS:

A react side-effect occurs when we use something that is outside the scope of React.js in our own react components E.g. Web APIs , localStorage, etc.

1. When we talk about side effects in the context of React.js , we are referring to anything that is outside the scope of react.

2.Making a HTTP request to an external API is another example of side effects.

3. We usually manage React side-effects inside the useEffect hook (part of the React Hooks).

HTTP METHODS

HTTP methods are a set of request methods to indicate the desired action to be performed for a given resource.

1. GET : The GET method is used to simply retrieve data from the server .
2. POST : The POST method sends data to the server for processing.
3. PUT : The PUT method is used to completely replace a resource identified by a given url.
4. PATCH : The PATCH method is used to partially update the resource.
5. DELETE : The DELETE method deletes the specified resource.

AXIOS

Axios is a promise-based HTTP library that lets developers make requests to either their own server or a third-party server to fetch data.

It offers different ways of making requests such as GET, POST , PUT/PATCH and DELETE.

MEMOIZATION

1. In Reatjs, performance optimization plays an important role in ensuring the smooth and efficient rendering of components. One powerful technique for optimizing performance is Memoization.

2. Memoization is an optimization technique that allows an increase in the performance of a program by storing the results of the function when the same inputs are given.

3. It prevents the unnecessary re-rendering of components.

PURE COMPONENTS

A react component is considered as a pure component if it renders the same output for the same state and props.

Its return value is always the same for the same input values.

React.memo()

1. React provides a built-in 'higher-order-component' (HOC) called as React.memo() that enables memoization for functional components.

2. The React.memo() wraps a component and checks for prop changes before re-rendering it.

3. If the props have not changed since the last render, React.memo() skips the re-rendering process and returns the previously rendered results, which is stored in cache.

useCallback Hook

1. useCallback is a hook provided by React that memoizes a function so that it can be reused across re-renders of a component without causing unnecessary re-renders.

2. When a function is defined within a component , it is recreated everytime the component is rendered. This can be problematic if the function is used as a prop to child components that use memoization, as the child components will always re-render even if the props they receive have not changed.

3. The useCallback hook takes two arguments: the function to memoize and an array of dependencies that the function depends on.

4. If any of these dependencies change , the function is re-created. If none of the dependencies change, the memoized function is returned.

5. Memoizing functions with useCallback can help improve performance in certain scenarios, such as when a component has a large number of child components that use memoization and receive function as prop.

6. It is important to be careful when using useCallback and make sure that dependencies array is set correctly. If the dependencies array is not set correctly , it can cause bugs and unexpected behaviour.

7. useCallback can also be used in conjuction with the useMemo hook to further optimize performance by memoizing expensive computations that are used as props to child components.

useMemo Hook

1. The useMemo hook is used to memoize a returned value.

2. It takes two arguments: a function and an array of dependencies.

3. This function is only executed when its dependencies change. If the dependencies don't change , the memoized value returned by the function is reused.

4. The second argument to useMemo is an array of dependencies. If these dependencies change, the function will be re-executed to produce a new memoized value.

5. useMemo is typically used to optimize the performance of expensive calculations or computations in a component. By memoizing these calculations , react can avoid unnecessary re-renders of the component.

***ROUTING (V5) ***

What is Router?

A router in ReactJs is a library that allows you to navigate between different pages or views within a single page application. It manages the UrLs of your application and maps them to appropriate components to render.

A router is needed to provide a seamless user experience when navigating through different parts of SPA.

How to use Router ?

To use a router in ReactJs, you need to install a router library such as react-router-dom using npm. Once installed , you can import necessary components from the library , such as BrowserRouter, Routes, Route, Link and use them in your React components to define the routes and navigation behaviour of your application.

** BrowserRouter**

It is a component that should wrap your entire application and provide the history and location objects to your components. The history object keeps track of the browser's history and the location object contains information about the current URL.

Routes

It is a component that wraps multiple Route components and renders Route that matches the current URL.

Route

It is a component that defines a route in your application . You can specify the path for the route and the component that should be rendered when the route is matched.

Link

Link is a component that provides declarative, accessible navigation around your application. It is used to create links to different routes and implement navigation around the application. It works like an HTML anchor tag.

useNavigate()

1. The useNavigate() hook is a built-in hook provided by React Router that allows you to navigate programmatically between different pages or components.

2. It returns a navigate function that can be called with the URL or a location to navigate to a different page or component.

3. To use this hook, you must import it from the react-router-dom library.

4. You can call the useNavigate() hook inside a functional component to access the navigate function.

5. The useNavigate can be used for navigating in response to user actions, such as button clicks or form submissions.

6.It is important to note that useNavigate should only be used inside a component that is rendered by routing , otherwise it will not work.

useParams()

1.The useParams hook is a hook that allows you to access parameters from the current url.

2.It returns an object containing key-value pairs of the parameters defined in the URL.

3. To use this hook, you must import it from the react-router-dom library.

4. You can call the useParams() hook inside a functional component to access the parameter values.

5.The parameter values can be accessed using the names defined in the url .

6. For ex. If you have a parameter named 'id' in your url , you can access it using

const {id} = useParams().

7. It is important to note that useParams should only be used inside a component that is rendered byrouting , otherwise it will not work.

Installation:

npm install axios , npm install react-router-dom

npm install react-icons

npm install json-server

npx json-server --watch backend/data.json --port 4000