**React Quick Prep Notes**

1. **What is React ?**

* React is a ui **library** whose only task is to render and create ui components.

1. **Is React a library or framework and what is difference between library and framework?**

* React is a library not a framework . Library is generally considered whose main function is just to complete one set of responsibility like the responsibility of the react is to create ui components.

Whereas the framework is the whole package with set defined of the utilities which we can only use .

1. **What are the features of React ?**

* The features of React are:-
  + **Declartive approach Rather than imperative approach**
  + **Component Based Architecture**
  + **Highly Flexible**
  + **Highly Performant due to virtual Dom**
  + **Easy Learning Curve**

1. **What are the cons of React ?**

* In my opinion there are some cons of react are:
  + Had to Make choice of Third Party Libraries
  + High Integration is Required
  + Much Freedom and Flexiblity which some time causes code bloating and following of non standard practices
  + Not a full fledged Framework

1. **What is JSX ?**

* JSX is syntax which is used in react to write components in easy way , and it somewhat similar to javascript with html written combined. JSX is also abbreviated as Javascript XML

1. **(Important) How JSX is send to the browser or what is BABEL?**

* Babel is transpiler whose task is to convert ES6 or react code to browser supported code . So before sending the JSX directly to the browser the Babel Transpiles the code to browser supported code and then re-renders

1. **What is the Difference Between the Element and Component ?**

* A **Element** is a piece of code which contains the logic (HTML + javascript ) component that will be rendered on the DOM Tree
* The **Component** is the Simple Javascript function which returns the Element. A function which returns the Element is Considered to Be Function

1. **What is the Syntax to Declare the Functional Component ?**

* Below is the Syntax to declare the react components
* const Test = () => {
* return (
* <div>Test</div>
* )
* }

1. **What are the ways to invoke the react Components?**

**🡪**The two ways to invoke the react components are :

* Invoke just like the function Invocation:- **component()**
* Using the Bracket Syntax :- **<Component/>(Preferrable and Most Used One)**

1. **What is Virtual DOM in react (Most Most Important)?**

* The speed of react rendering of the component lies in the virtual DOM , A Virtual DOM is actually a snapshot of the actual DOM which is currently being rendered on the page . While Updating the component react Matches the actual dom with virtual dom to find which parts of the component is changed , using the **diffing algorithm . And then it only updates that component in real dom which is changed . Which increases the speed and performance.**

1. **What is React Fiber and React Reconciliation?**

* React Reconciliation is Feature where the React sees the changes ,changes the virtual dom and using the diffing algorithm changes the final dom to render the component.

1. **What is Difference between the react and reactDom library ?**

* When we install the react the react and react-dom two packages are installed .
  + **React Package –** The React packages contains all the core feature of the react like useState,useEffect and other features .
  + **React-Dom Package –** The React Dom package is specified to attach react to the web browser , The react dom first create Root Element from the dom Element and under that Root Element whole React Tree Renders.

**State vs Props:- Most Most Impotant**

One of the most asked and one of the most topic is what is the Difference between State and Props.

1. **What is Difference between the state and Props?**

* Lets Start with the State:-
  + **State –** State is the variable local to the component and which store the snapshot of the data at the particular instance of time . Its Actually a variable which is under the observation of the react . The way to change the change is from the setter Function which is attached to the state variable , whenever the state changes the component will re-render
  + **Props –** The Props are the variables which are used to pass the data from one component to another component. **Props are only read only type .** The change of the props doesn’t guarantee that component will re-render.

1. **What are the ways in which the component will re-render?**

* The component will re-render when the
  + **State of the component changes , includes nested States also.**
  + **Parent Component is Re-Rendered .** If the parent component is re-rendered than all its child component is also re-rendered

**React Hooks 🡪 The Core of the React**

The react is using the functional components and react core lies in its hooks , the hooks are the power of the react which helps in the re-rendering and side effect handling in the component

Below is the list of the react hooks which are super important and we will learn about them in details

* useState
* useEffect
* useRef
* useMemo
* useCallback
* useContext
* useLayoutEffect
* memo
* createContext
* useReducer
* useImperativeHandle

1. **What is the useState Hook?(Most Important)wh**

* The useState hook is most basic and primary hook which is used to create a state variable and its setter function.
* Below is the syntax of the state hook

 const [state,setState]=useState('InitialState');

* The Component always re-renders the component when the component state changes the comparison is based On **Object.is method** so for object and array we should always update the state with always creating new object and new array.

1. **What is useReducer Hook?(Most Important)**

* The useReducer Hook is also used to update the state, but using useReducer we can handle the complex state updation in precise and easy way . using the reducer function which is the pure function . we dispatch the actions using the dispatcher which is returned from the useReducer hook and reducer function will update the state based on those actions.

Below is the syntax for the useReducer Hook

*// useReducer hook*

    const [state, dispatch] = useReducer(reducer, initialArg,init)

In the above syntax Lets under Each Component

* + 1. **Reducer –** It is the pure function which take **state and action** as the argument and which will listen to action and based on that update the state and return new state
    2. **initialArg-** This is initial state value we need to provide for state
    3. **state –** Its is the current snapshot of the state
    4. **dispatch –** This is dispatcher function which will dispatch the action which reducer will listen to .

1. **What is useRef Hook ?(Most Important)**

**🡪**UseRef hook helps to reference a variable or component

**Some important properties about the useRef Hook**

1. useRef hook only has one property assigned to it which is **current Property.**
2. useRef reference variable **does not change during re-rendering**
3. The Use Ref Hook promotes the imperative structure of code practice where we can manually access the properties of the element

**useRef Caveats**

1. We should not change the useRef.current property on re-renders , we should only change the property of the useRef.current property inside the functions or in the useEffect()
2. When attached to the element the useRef should be used to read the properties of the component , it is not advisable enough to change the properties of the with useRef , we can change the properties but not a good practice
3. **What is the ref on the elements Property ?(Important)**

* In react every native dom element we create has access to the ref property we can assign the our useRefed reference variable to this react dom property and then we can access all the properties of that element

1. **What is the forwardRef in the React ? (Most Important)**

* So every native element of the dom has access to the ref property but what about the custom Components we create how to pass the ref property on them , **cant we just pass the ref property as just the props. The Answer is no**
  + **Forward Ref ->**the Forward Ref is the higher order function which takes **two argument first is the custom component we have created and other is the** ref which we want to attach or pass into our custom component
* import { forwardRef } from 'react';
* const MyInput = forwardRef(function MyInput(props, ref) {
* *// ...*
* });

1. **What is the useEffect Hook ? (Most Most Important concept)**

* React is only UI library , whose only responsibility is to handle and render the ui components . But in UI development there are other side Effect like calling the api and doing other stuffs apart from rendering the UI components.

So there is a hook called **useEffect** which is effectively used to handle these side effects only . Now Lets Learn about the basic and advance features of useEffect.

Important points of useEffect hook:-

* + Below is the syntax for the useEffect hook

useEffect(()=>{

*// some logic*

},[])

Parts:-

()=>{}// callback function

[]// Dependency Array

* + The useEffect Hook takes two parameters**: A Callback function and an dependency array .**
  + **(Important)** The useEffect only runs after the component is rendered , hence the useEffect is non- blocking for rendering purpose

1. **What is the Dependency Array in the useEffect hook ?(Most Most Important)**

* In the useEffect Hook the dependency array is super important , this dependency array dictates and tells at when the callback function will be invoked
  + **useEffect Hook with dependency Array ->** The useEffect hook without the dependency array in this scenario the callback function **will be invoked in every re-render** process (**We should never use this in practical scenario** )
  + **useEffect Hook with Empty Dependency Array 🡪**The useEffect with empty dependency Array only invoke the callback function only once as soon as the content is loaded first time , after the first component render , the useEffect will not invoke the callback function(**Should be Used Mostly)**
    - **Example use Cases – One Page Load we have call api**
  + **useEffect Hook with Filled Dependency Array 🡪** The useEffect with Filled Dependency Array , will reinvoke the callback function every time the state variable in the dependency array changes. We can pass multiple dependencies and even if the single dependency changes the callback function will be re-invoked

1. **what is useEffect clean up function ?(Most Most Important)**

* So the useEffect has one optional return function into our callback function

Below are some important properties of that function

* + The **clean up function will run every time before the invocation of actual callback function , whenever the dependencies changes**
  + **The clean up function will always run when the component is dismounted**
  + **The clean up function will not run on the first render of the component**

Below Is the syntax for the clean up function with useEffect

*// use Effect with cleanup function*

useEffect(()=>{

*// some logic*

    return ()=>{

*// cleanup logic*

    }

},[])

1. **What is the useLayoutEffect hook?**

* useLayoutEffect is hook which is pretty much similar to the useEffecthook but the only difference is **useLayoutEffect Hook is synchronous in nature .** 
  + **Before** every re-render the useLayoutEffect Hook will run before the component is loaded

**Important Note :-**

**As the useLayoutEffect is synchronous is nature we must catiously use it as has performance issues and can cause rendering freeze if the useLayoutEffect callback function is performance heavy**

**Optimization Hooks**

1. **What is React.memo hooks and how it works?**

**🡪**We know that when the parent component re-renders due to state changes , all its child component will also be rendered

**🡪**.**Due to some cases the components which are not dependent on that state change will also re-render . This is problem and can lead to performance issues especially if the child which is rendering has deep sub child trees**

* To prevent the above scenario we react provides us with memo function
* Memo function is HOC function which wraps the component which need to memoized
* What **memo function does is it keeps check on the props of the component and only re-renders the component only if the props changes and if props doesn’t change it will not re-render even if the parent is re-rendered**
* Below is the syntax of the memo function
* *// memo function example*
* const MemoizedComponent = memo(SomeComponent, arePropsEqual?)
* As seen in the above syntax the memo function two arguments which are
  + **Component –** The component which needs to be memoized
  + **arePropsEqual(Optional)-** We can also provide our custom function to provide the comparison logic between our components

**Important Note:-**

* Using the memo doesn’t necessarily means the performance will definitely increase because comparison of props each time is also expensive , A developer needs to choose between when to memo the component and when not .
* Should not memoize each and every component
* **Even if the component is memoized it will re-render if the its inner state variables the memo only cares with the props changing**

1. **What is useCallback Hook ?(Most Most Important)**

* The useCallback functions are used to control the **creation of the functions in React Component**
  + It is generally used when we pass the instance of the function from parent to child
  + **Now for every re-render the new instance of the function is created and if the function is created then , props has changed .even the child component is wrapped with the memo HOC even then the component will render**
  + Below is the syntax for the useCallback Hook
* *// Use Callback function Syntax*
* const cachedFn = useCallback(fn, dependencies)

**lets** see the various parameters of the useCallback hook

* + **fn –** This is the function we need to memoize , with useCallback for every render the useCallback will return the same instance of the function ie memoize function
  + **dependencies:**-This is the dependencies array which when changes only then the new instance of the function is created .(**Very Very Important for preventing bugs)**

1. **What are caveats in the useCallback function?**

* One of the caveats of the useCallback which is its **freezing of the function problem**
  + As we know that useCallback returns the memoize function , what it does is it takes the snapshot of all the properties in function at the first render time .
  + This can cause bugs as the same freezed properties value is returned , even if the first properties values has been changed
  + **So we always have to take care by adding the dependencies to useCallback function . most most Important thing**

1. **What is the useMemo Hook (Most Most Important another performance optimization hook) ?**

* useMemo hook is another performance optimization hook , which caches or **memoize the expensive function invocation.**
  + This hook memorizes the expensive function invocation , which occurs on every render .
  + This hook helps to optimize the local component performance if which is the function is invoking
  + Below is the syntax of the useMemo
* *// use memo hook syntax*
* const cachedValue = useMemo(calculateValue, dependencies)

Lets see the component of the useMemo syntax

* **calculateValue:**This the value from which is result expensive function invocation which we need to memoize
* **dependencies:**This is the dependencies array which will govern when the function invocation will reoccur, whenever the function dependencies changes the function will invoke and return the new value

1. **What are the caveats of the useMemo (Most Most Important)?**

🡪 The useMemo hook is the complex hook if not used properly it can create a lots of bugs .

* Lets see the caveats of the useMemo hook
  + **The Freezing Problem ->**As we know the when we pass the calculated value the useMemo freezes that value snapshot at the first render and continue to use it
  + **To Resolve the Above problem** we need to add the dependencies so whenever the dependencies changes the value should be re-calculated so that updated value is being served.

**A Multi Component State Provider Hook – useContext(Very Very Important)**

1. ­**What is the useContext Hook (Most Most Important)?**

* We know that react is mostly about the sharing state to multiple components.
  + We can easily provide the state to the direct parent from child and then its child component – Known as **Prop Drilling**
  + But what about how to share the state from detached components which are lying in different trees .
  + **This Problem is solved by the hook useContext , by which we can share the state to multiple components along the whole React Dom**

Definition🡪useContext is a React Hook that lets you read and subscribe context from your component.

**The useContext Hook has three Main Component which creates the whole context**

**🡪createContext**

**🡪useContext**

**🡪Provider and Consumer(class based)**

**Below is the Syntax of the useContext hook:-**

*// useContext hook Syntax*

const someContext = createContext(initialValue)

const value = useContext(SomeContext)

<someContext.Provider value={value}>{components which need to provide values}</someContext.Provider>

Lets see the each components of the above syntax

* **createContext ->**This is the react provided function which will create the Context with the initial values which we have provided and returns that context
* **Provider ->** the provider task is took provide the context to all the components which are wrapped under the provider , only the components which are wrapped under the provider can use the context value which can be accessed using the useContext hook
* **useContext->** The useContext hook takes that context object and returns the states which we have passed using the providers

**Some Other Less Used Hooks But Important**

🡪useImperativeHandle

🡪useDeferredValue

🡪useTransition

1. **What is the useImperativeHandle hook ?**

**--->**useImperative handle hook is the hook which solves the unique scenario generally some times we want some call a function store in the child component and we have call the the function in the parent component

---> Yes we can also pass the reference function from parent to child and then invoke the function from child and parent will execute the logic this is most appropiate way

---> but Sometimes the Problem is that some third party function is their so we want to expose only **some function of the dom like focus or scroll to view . for that scenario we use the useImperative Hook**

* **In short lifting the state up using the refs**

**Below is the syntax of the useImperativeHandle hook**

 // useImperative Handler

    useImperativeHandle(ref, () => {

        return {

            reset

        }

    },[])

**React Router DOM**

1. **What is the react Router Dom ?**

🡪React Router DOM is the client side routing library which helps us to create routes within our applications .

🡪The react router Dom provides various Component which are use to create the Routing in the react application,

1. **What are Various Component which are required create the Routing?**

**🡪BrowserRouter->**This is the Top Wrapper Component which we need to wrap our whole application on which we need to implement the Routing Logic.

Below is the Example of the **BrowserRouter Wrapping the component . There are multiple Router which react Router Dom has been provided.**

<BrowserRouter>

      <Provider store={store}>

        <App />

      </Provider>

    </BrowserRouter>

**🡪Routes and route ->** The Routes and Route are function the which are used to define the routes.

<Routes>

          <Route

            path='/\*'

            element={<CineSearchPage />}

          />

          <Route

            path='/:id'

            element={<MovieDetailsPage />}

          />

          <Route

            path='/login'

            element={<LoginPage />}

          />

          <Route

            path='/sign-up'

            element={<SignUpPage />}

          />

          <Route

            path='/my-watch-list'

            element={

              <ProtectRoute>

                <WatchlistPage />

              </ProtectRoute>

            }

          />

        </Routes>

Below is the example of the How to Declare the Routes

**Question 3: What are the different types of Router Present in react router dom?**

* The Router Provides us with various Router Objects which are:
  + BrowserRouter
  + HashRouter
  + StaticRouter
  + Memory Router

**Question 4) What are the dynamic Routes and how to handle the dynamic Routes**

* The Dynamic Routes or the route-params are the route parameters which in combination with base route create a unique route

Below are the example of the route params

🡪http://localhost:3000/blog/**1 🡪 here 1 is dynamic Parameters**

**How to declare the Dynamic Route in the react router dom**

**🡪** <Route

            path='/:id'

            element={<MovieDetailsPage />}

          />

**Question 5)How to access the path params of the current route in our code ?**

* We can use the useParams() hook provided by the react router dom to access the path params . useParams hook provide the object with the path parameter
* const params=useParams()
* where the params is the object whose keys are the path Variable holder and its value is the actual path variable value

**Question 6)**

**Advance React Questions for the Interview**

1. **What are the performance improvement techniques you have used in react applications?**

**Answer-** Various Performance optimization technique which can are used in the react applications are :

1. Use of performance optimization hooks like **useMemo and useCallback** to cache expensive values and function recreation
2. Memoize your components using **React.memo**
3. **Keep your components as lean and as pure as possible**
4. Using lazy loading of the components and using CDN to load your images
5. **Tree shaking and removing of the dead code , optimizing your imports**
6. Bundling and chunking your code js files to small small chunks which are loaded just on requirements
7. Avoid using index as key’s for react application
8. Do not spread all props on the div elements