**REACT JS INTERVIEW QUESTION**

1. **WHAT IS PROPS IN REACT ?**

In React, "props" is a shorthand for "properties,". it's an essential concept used for passing data from a parent component to a child component to render on UI.

The data received as a props by child component its a read-only data. they can’t modified this data, the parent component is responsible for updating and making changes to this data.

EX-

/ ParentComponent.js

import React from 'react';

import ChildComponent from './ChildComponent';

const ParentComponent = () => {

const name = 'John Doe';

const age = 25;

return (

<div>

<ChildComponent name={name} age={age} />

</div>

);

};

export default ParentComponent

// AT ChildComponent.js

import React from 'react';

const ChildComponent = (props) => {

return (

<div>

<h2>Name: {props.name}</h2>

<h2>Age: {props.age}</h2>

</div>

);

};

export default ChildComponent;

1. **what is the Component in react ?**

In React, a component is a reusable piece of code that defines the structure and behavior part of a user interface and ItS returns the HTML element.

Components are the building blocks of React applications, multiple component create Complex UI .

There are two types of components in React: functional components and class components.

Functional Components: Functional Components are simple JavaScript functions that accept props (short for properties) as input and return JSX (JavaScript XML) elements. They are also known as "stateless components" . because they don't have their own internal state management. Function components are easier to read, test, and maintain.

EX-

function Greeting(props) {

return <h1>Hello, {props.name}!</h1>;

}

1. **What is REACT ..?**

React is an free open-source JavaScript library used for building user interfaces (UI) or front-end web applications. . It was developed by Facebook and is currently maintained by Facebook, Instagram, and a community of individual developers. . React was first released in 2013, and it has since gained immense popularity due to its efficiency and ease of use.

The main purpose of React is to Create a reusable UI component that can efficiently update and render based on changes to the underlying data. It follows a basically component-based architecture, where the website UI is broken down into small part, and for each part e create a separate component and Each component can manage its own state and properties, which makes it easier to build complex user interfaces in a structured and maintainable way. React uses a virtual DOM (Document Object Model) to optimize rendering performance. Instead of directly updating the actual DOM when a change occurs.

1. **What is the useState Hook?**

UseState hook is an inbuilt hook in React. It allows to use a state in react . the useState hook returns an array with two elements. The first one is State and the second one Function. the state variable which is used to store the initial state of value. and the function is used for make changes and updates in this state.

Ex-

import React, { useState } from 'react';

function ExampleComponent() {

// Declare a state variable named 'count' with an initial value of 0

const [count, setCount] = useState(0)

return (

<div>

<p>Count: {count}</p>

<button onClick={ () => setCount(count+1) }>Increment</button>

</div>

);

}

1. **What is UseCallback in React?**

The usecallback hook is another built-in hook in React that is used to optimize the performance of functional components, When you create a function inside a functional component, t, it is recreated every time the component is rendered.. If you pass this function as a prop from parent component to the child component, then another child component is also re-rendering unnecessary where function is actually has not been passed, wch make it decrease the performance of react application.

The use callback hook allows you to memoize the function and return a memoized version of it, so the function reference remains stable across renders, as long as the dependencies haven't changed.

1. **what is useMemo hook in React ?**

The useMemo hook is another built-in hook in React that is used to memoize the result of a computation and cache it.

When we create or deal with a multiple functions and one function is connected to another function.

1. **What is useEffect Hook in reactjs ?**

useEffect is a hook in React that allows you to perform side effects in functional components. Side effects can include things like fetching API data, setTimeOut, setInterval, manual manipulating the Dom, and more.

**16) what is Prop Drilling in React?**

Prop drilling in React refers to the process of passing down data as a props from the parent component to child component then another child component and so on until it reaches that specified component that actually need data, in this case unnecessary data pass to that component who don’t need. Its follow a nested level of tree. So, its become very complex and readability weak.

Prop drilling can make the codebase harder to maintain, less readable, and less efficient.

To Solve the issues of prop drilling, you can consider using other state management techniques such as:

Context API: React's Context API allows you to create a global state that can be accessed by any component without the need for prop drilling.

State Management Libraries: Libraries like **Redux** or MobX provide centralized state management that can eliminate the need for prop drilling.

Custom Hooks: You can create custom hooks to manage specific pieces of state or logic, allowing you to access the state directly without passing it through multiple components.

**7)What is ContextAPI in React?**

Context API is a feature in React, which provide a feature to pass data from the parent component to the child component directly instead of passing data from each component level of the tree to that component that actually needs data and share data between components without having to pass props through multiple levels of the component tree. Instead of manually passing data down through the component hierarchy,

To use context Api we can create first –

Create Context, then provider context and <data.consume/> tag use on that child who need data.

File - contextApi.js

**8)What is useContext Hook ?**

useContext is a hook in React that allow passing a data from the parent component to child component directly bu use of useContext Hook. Context provides a feture to pass data between components without having to pass the data explicitly through props at every level of the component hierarchy or tree .

Here's a basic overview of how useContext works: Create a Context: First, you create a context using the React.createContext() method.

Provide a Value: then data.provider tag use , You wrap a part of your component tree and their value wch we have to pass in the Provider component created tag This provider provide a value that you want to share with the components.

Consume the Context: Instead of passing props down through every intermediate component, you can use the useContext hook to directly access the value provided by the Provider.

File – useContextHook file

9)**what is Jsx in React?**

JSX, which stands for "JavaScript XML,. It is a syntax extension for JavaScript file which is commonly used in the React library for building user interfaces. It allows developers to write HTML-like code within JavaScript file, which making it easier to define and render UI components in a more readable and intuitive manner .

There are some rule when you write a code in Jsx file –

1. ALL code must be wrapped into top level Element .
2. We do not use if else conditional operator instead of this we use Ternary operator.
3. All JSX expressions must be written in curly braces {}.
4. We can use class coz class is a reverse keyword so we use instead of this is ClassName.
5. **10)** **What is Axios in React?**

Axios is a popular JavaScript library that is used for making HTTP requests from web browsers on the server or Node.js. It is used to send and receive data from APIs or other servers. In the context of React, Axios is often used to fetch data from APIs and update the state of React components based on that data.

We use Axios. get for fetch API.

Some key features of Axios include:

Promise-Based API: Axios uses promises for handling asynchronous operations. This makes it easier to manage asynchronous code and handle responses in a more readable and structured way.

Browser and Node.js Compatibility: Axios can be used both in web browsers and Node.js environments. This makes it a versatile choice for making HTTP requests in various settings.

Interceptors: Axios allow you to define interceptors that can be used to globally handle requests and responses. This can be useful for tasks like adding authentication headers or handling errors consistently.

Cancellation: Axios support request cancellation, which can be helpful in scenarios where a user navigates away from a page or an API call becomes unnecessary.

Request and Response Transformation: axios enables you to transform requests and responses before they're handled by your application. This can be useful for formatting data or modifying headers.

**//Fetch Api**

Fetch: The Fetch API provides a fetch() method defined on the window object. It also provides a JavaScript interface for accessing and manipulating parts of the HTTP pipeline (requests and responses). The fetch method has one mandatory argument- the URL of the resource to be fetched. This method returns a Promise that can be used to retrieve the response of the request.

fetch('path-to-the-resource-to-be-fetched')

.then((response) => {

// Code for handling the response

})

.catch((error) => {

// Code for handling the error

});

**By Axios Method**

axios.get('url')

.then((response) => {

// Code for handling the response

})

.catch((error) => {

// Code for handling the error

})

**11)** **WHAT is REDUX ?**

Redux is an open-source JavaScript library or predictable state container which is used for managing the state of applications, It provides a predictable and centralized way to manage and update application state, making it easier to reason about how data changes over time.

In Redux we create a Slice.jsx file

In slice, we define –

Initial state

Name of state

Reducer function.

Some Key concepts in Redux:

Store: it is a centralized store file of Redux which holds the entire state tree of your website application. Or where multiple slice file store. It's a JavaScript object that represents the current state and provides methods to read the state, dispatch actions, and subscribe to changes.

Actions: Actions are plain JavaScript objects that describe an event or change in the application. They have a type field that indicates the type of action and additional data if needed.

Reducers: Reducers are functions that specify how to make changes in state value. Each reducer is responsible for managing a specific part of the state and returns a new state object without modifying the original state.

Dispatch: Dispatching an action is the way to trigger a state change. When an action is dispatched, Redux invokes the appropriate reducer(s) and updates the state accordingly.

12**) What is useSelector Hooks ?**

The useSelector hook is a feature provided by the React Redux library, which is commonly used in conjunction with React to manage the state in a Redux application.

It provides a feature to access and retrieve data from the Redux store within a React functional component.

In Redux, the state of an application is stored in a central store, and components can access this state using the useSelector hook. The useSelector hook allows components to select and extract specific pieces of state from the Redux store, based on the provided selector function.

It receives the entire state as an argument and returns the desired data. When the selected data changes, the component using useSelector will be re-rendered.

Ex –

import React from 'react';

import { useSelector } from 'react-redux';

const MyComponent = () => {

const selectedData = useSelector(state => state.myReducer.selectedData);

// Rest of the component logic

return (

<div>

{/\* Render the selected data \*/}

<p>{selectedData}</p>

</div>

);

};

export default MyComponent;

**13 ) what is state in React ?**

In React, "state" refers to the data which is managed by component and it can change or updated over time. It represents the intial data or current of the component which used for building interactive and dynamic user interfaces of website.

State allows you to store and track data that can be modified and will trigger the component to re-render when it changes.

You can initialize state within a component using the useState hook (in functional components) or **this.state** object (in class components).

A basic overview of how state works in React:

Initializing State -

Ex-

import React, { useState } from 'react';

function Counter() {

const [count, setCount] = useState(0); // Initial state value of 0

}

Updating State –

function Counter() {

const [count, setCount] = useState(0);

const increment = () => {

setCount(count + 1); // Update the state with a new value

};

}

Rendering with State:

function Counter() {

const [count, setCount] = useState(0);

return (

<div>

<p>Count: {count}</p>

<button onClick={increment}>Increment</button>

</div>

);

}

**14) What is Custom Hook?**

A custom hook in React is a JavaScript function that allows you to reuse a piece of code or stateful logic which is used in across multiple components. It's a way to extract common logic from components and write this logic with the help of function in a separate file that is known as Custom Hook file and then share it in a multiple components in a reusable manner. Custom hooks follow a specific naming convention, where the name starts with the "use" keyword.

A custom hook in React is a JavaScript function or allows to use utilizes the built-in React hooks (like useState, useEffect, useContext, etc.)

Custom hooks enable you to encapsulate complex logic and provide a clean and reusable interface for other components to consume. They can be used to handle state management, side effects, subscriptions, and more,

Ex- Random gif project.

**15) what is State lifting?**

State lifting, also known as "lifting state up,". it is a process where pass data from the child component to PARENT Component directly in React. In other words, we can also say it is a pattern that moves the management of a piece of state from a lower-level component to a higher-level component.

This is done to share and synchronize the state across multiple components because they need access to the same data.

State lifting is used when multiple components need to share same state so Instead of keeping or duplicating the same state in multiple components, we centralize the state management in a common ancestor component.

EX-

**import React, { useState } from 'react';**

**// A child component that receives a value and a function to update it**

function ChildComponent(props) {

return (

<div>

<p>Value: {props.value}</p>

<button onClick={props.onIncrement}>Increment</button>

</div>

);

}

// The parent component manages the shared state

function ParentComponent() {

const [count, setCount] = useState(0);

const handleIncrement = () => {

setCount(count + 1);

};

return (

<div>

<ChildComponent value={count} onIncrement={handleIncrement} />

</div>

);

}

function App() {

return (

<div>

<ParentComponent />

</div>

);

}

export default App;

**16) what is JQuery?**

jQuery is a fast, small, and feature-rich JavaScript library. It simplifies various tasks related to HTML document manipulation, event handling, animation, and Ajax interactions. jQuery makes it easier for developers to write concise and efficient code by providing a unified and simplified API for working with the Document Object Model (DOM) and handling common web development tasks.

Some of the key features of jQuery include:

DOM Manipulation: jQuery provides a convenient way to select and manipulate HTML elements on a web page. You can easily change content, attributes, and styles of elements using simple commands.

Event Handling: jQuery simplifies event handling by providing methods to attach event listeners to elements. This enables you to respond to user interactions like clicks, mouse movements or hover ,and keyboard inputs.

Animations: jQuery includes functions for creating smooth animations and transitions on web pages. You can animate properties like width, height, opacity, and more.

Ajax Interactions: jQuery makes it straightforward to perform asynchronous HTTP requests (Ajax) to retrieve or send data to a server without requiring a full page reload. This is commonly used for building dynamic web applications.

Cross-browser Compatibility: jQuery handles the differences between various web browsers, providing a consistent programming interface regardless of the browser being used.

Plugin Support: jQuery's plugin architecture allows developers to extend its functionality with third-party plugins. This has led to the creation of a vast ecosystem of plugins for a wide range of purpose