**1.What is React?**

open-source front-end JavaScript library developed by Facebook.

It follows the component-based approach for building reusable UI.

**2. Features and Advantages of React:**

***less coding*** and offers more functionality.

React uses ***Virtual DOM***, thereby creating web applications faster.

***Reusable components***.

React follows a ***unidirectional*** data flow.

***easy to learn***, as it mostly combines basic HTML and JavaScript concepts.

It can be used for the development of ***both web and mobile apps***.

**server-side rendering.**

**3. functional** **components VS class components.**

**Functional components** are simple JavaScript functions that take props as arguments and return JSX.

A **class component** extends from React.Component and the render function return the React element.

To use properties available in React in your class component you must inherit the Component class.

**Functional** **components Class** **components**

NO render method. It must have the render() method returning JSX.

Stateless components Stateful components.

React lifecycle methods cannot be used. Supports React lifecycle methods.

Constructors are not used. Constructor is used as it needs to store state.

Simple More Complex

useState()Hook for handling state this keyword is required, along with the setState()

function and a constructor.

**4.Virtual DOM:**

The Virtual DOM is copy of the Real DOM. It optimizes and speed up the process of updating the real DOM by comparing the current virtual DOM with the previous one.

Real DOM manipulation is slower than virtual DOM manipulation.

When the state of an object changes in a React application, VDOM gets updated. It then compares its previous state and then updates only those objects in the real DOM instead of updating all the objects.

**5.Virtual DOM working**

1. Whenever any data changes in the React App, the entire UI is re-rendered in Virtual DOM representation.

2. Now, the difference between the previous DOM representation and the new DOM is calculated.

3. Once the calculations is completed, the real DOM updated with only those things which are changed.

**6.JSX:**

JSX stands for JavaScript XML.JSX allows us to write HTML in React. JSX is an extension of the JavaScript language based on ES6 and is translated into regular JavaScript at runtime.

**7.How browser works with JSX?**

Browsers cannot read JSX directly because they can only understand JavaScript objects, and JSX is not a regular JavaScript object. Thus, we need to transform the JSX file into a JavaScript object using transpiler (high-level source code to another high-level) like Babel and then pass it to the browser.

**8.React vs Angular**

**React Angular**

JavaScript-based library. TypeScript-based full-fledged framework.

Virtual DOM Real DOM

one-way data binding two-way data binding

speed faster slower as compared to react.

**9.Components:**

In React, components are the building blocks of React applications. These components divide the entire React application's UI into small, independent, and reusable pieces of code.

**10.Props:**

Props stand for properties. It gives a way to pass data from the parent to the child components throughout the application.

Props are like function arguments in js and attributes in HTML.

Props are an object which stores the value of attributes of a tag.

Props are immutable i.e. once set the props cannot be changed.

Props can be accessed by child components.

**11.State:**

State is an object that is to be used to hold/store the data that may change over time.

State is generally updated by event handlers.

States can be used in Class Components, Functional components with the use of React Hooks (useState and other methods) while Props don’t have this limitation.

State cannot be accessed by child components.

**12.Why should we not update the state directly?**

It won't re-render the component. Instead use setState() method.

**13.What is render() function?**

render() method is used to return the HTML which you want to display in a component.

**14.Stateless vs Stateful:**

Stateless Component Stateful Component

do not hold or manage state. can hold or manage state.

It is also known as a functional component. It is also known as a class component.

It is simple and easy to understand. It is complex as compared to the stateless

It doesn’t work with any lifecycle method It can work with all lifecycle method.

The stateless components cannot be reused. The stateful components can be reused.

Behavior is independent of its state. Behavior is dependent on state of comp.

**15.What is SPA?**

A single-page application is an app that doesn’t need to reload the page during its use and works within a browser.

**16.Controlled vs Uncontrolled component**

Controlled Uncontrolled

1. Usage of Component State is a must. Usage of Ref is a must.

2. form data is handled by a React component. form data is handled by the DOM itself.

3. It accepts its current value as a prop. It uses a ref for their current values.

4. It allows validation control. It does not allow validation control.

5. better control over the form elements and data. limited control

**17.Why component names start with a capital letter?**

component names start with a capital letter else it will throw an error as an unrecognized tag. It is because, in JSX, lower case tag names are considered as HTML tags.

**18.React Hooks:**

Hooks are built -in react functions that allow to use React features such as state, lifecycle method and context. Because of this, class components are generally no longer needed.

**19.Rules of Using React Hooks:**

Hooks can only be called inside React function components.

Hooks can only be called at the top level of a component.

Hooks cannot be conditional.

**20.State Lifting up:**

Pass data from child component to parent component.

Create function in parent component. call function in child component with data as parameter.

**21.useState() Hook:**

The React useState Hook allows us to manage state in a function component.

The useState hook is a special function that takes the initial state as an argument and returns an array with two values: the current state and a function to update state value.

**22.UseEffect Hook :**

The useEffect hook lets you manage side effect in functional components. side effects are fetching request, manipulating the DOM, and timers.

The useEffect accepts two arguments a callback function and dependencies. The callback function contains the side effects, while the dependencies are optional.

To run useEffect on every render do not pass any dependency. To run useEffect only once on the first render pass any empty array in the dependency.

**23.useRef Hook :**

It can be used to store a mutable value that does not trigger a re-render when updated.

It can be used to access a DOM element directly.

It is a built-in React hook that accepts one argument as the initial value and returns a reference object having a special property current.

**24.Reference vs State:**

Updating a reference doesn't trigger re-rendering, while updating the state makes the component re-render;

The reference update is synchronous (the updated reference value is available right away), while the state update is asynchronous (the state variable is updated after re-rendering).

**25.Context API:**

Provides a way to pass data through the component tree without passing prop to every level.

**26.useContext** hook enables you to access the context data without using Consumer component explicitly.

**27.forwardRef:**

By using forwardRef, you can pass a reference from a parent component to a child component, even if that child component is wrapped inside another component. This enables the parent component to interact with the child’s DOM element or instance directly.

**28.useMemo hook:**

It is very useful in optimizing the performance of a React component by eliminating repeating heavy computations and avoid the unnecessary re-rendering of components.

The React useMemo Hook returns a memoized value. Memoized value does not need to be recalculated. It stores each computation and return the same value when required again.

useMemo hook takes two arguments: the calculation function and an array of dependencies.

The useMemo Hook only runs when one of its dependency’s updates.

**useCallback Hook useMemo Hook:**

Returns a memoized callback.Returns a memoized value.

returns its function uncalled when

the dependencies change. calls its function and returns the result.

**Higher-order components (HOCs):**

HOCs are a powerful feature of the React library. They allow you to reuse component logic across multiple components.

In React, a higher-order component is a function that takes a component as an argument and returns a new component with additional functionality.

**React router & Routing:**

A React router is a library of React, which supports the routing mechanism. If We want to create an application which contains multiple pages, and we want to navigate to those pages without reloading the webpage, that is only possible with the help of a react router.

.forEach(), is used to execute the same code on every element in an array but does not change the array and it returns undefined.

.map() executes the same code on every element in an array and returns a new array with the updated elements.

.filter() checks every element in an array and returns a new array with the elements that return meets a criterion.

**Lifecycle of Components:**

Component creation process which involves various lifecycle methods are termed as component lifecycle.

In React, components have a lifecycle that consists of different phases. Each phase has a set of lifecycle methods.

The three phases are: Mounting, Updating, and Unmounting.

React lifecycle methods are predefined functions allow you to control the component's behavior and perform specific actions at various phases of a lifecycle.

**1.Mounting:**

The mounting phase refers to the phase during which component is created & inserted to the DOM. The mounting phase has three main lifecycle methods that are called in order.

1***.constructor( )*** method is called before the component is mounted to the DOM.

It used to initialize state and bind event handlers methods. You should not call setState( ) in the constructor( ).

2.***getDerivedStateFromProps( )*** method is called right before rendering the element(s) in the DOM.

It is a place to set the state object based on the initial props.

It takes state as an argument and returns an object with changes to the state.

3.***render()*** is responsible for rendering the component's JSX representation onto the screen. It returns the JSX code that describes what should be displayed. When the component file is called it calls the render() method by default. If state or props have changed the component is re-rendered.

4.***componentDidMount( )*** method is called after the component is rendered.

You may call setState( ) immediately in componentDidMount( ).

**2. Updating:**

A component is updated whenever there is a change in the component's state or props.

***componentDidUpdate:***

The componentDidUpdate method is called after the component is updated in the DOM.

You may call setState( ) immediately in componentDidUpdate( ) but note that it must be wrapped in a condition else result in an infinite loop.

**3. Unmounting:**

This method is called when a component is being removed from the DOM.

***componentWillUnmount( ):***

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

It is used to perform any necessary cleanup.

You should not call setState( ) in componentWillUnmount( ) because the component will never be re-rendered. Once a component instance is unmounted, it will never be mounted again.

**Fetch API in React**

it doesn't require additional libraries or packages to be installed.

Web APIs is used to fetch data from a database and save data back to the database.

GET: Used to request data from an endpoint

POST: Sends data to an endpoint

DELETE: Remove data from an endpoint.

PUT: Update a record or data value at an endpoint.

Ways of Fetching Data from API

There are different of fetching data:

By using Fetch API

By using Axios library

By using async-await syntax

**1.Fetch API**

We can fetch data by using JavaScript fetch( ) method. It will request sever and load the information on the web pages. It will return a promise.

it doesn't require additional libraries or packages to be installed.

**GET:**

const [user, setUser] = useState([ ]);

const fetchData = ( ) => {

return fetch("url")

.then( (response) => response.json( ) )

.then((data) => setUser(data));

}

useEffect(( ) => { fetchData( );}, [ ] )

**DELETE:**

useEffect(( ) => {

fetch( `https://jsonplaceholder.typicode.com/posts/${id}`,

{ method: 'DELETE' })

.then( (response) =>{ response .json( ).then

( ( data ) =>{ cosole.log( data); fetchData( );})

})

}, [ ]);

**POST:**

useEffect(( ) => {

const requestOptions = {

method: 'POST',

headers: { 'Content-Type': 'application/json' },

body: JSON.stringify({ title: 'React Hooks POST Request Example' })

};

fetch(‘url’, requestOptions)

.then(response => response.json( ))

.then(data => setPostId(data.id));

}, [ ]);

**PUT:** same as DELETE just add header and body.

**2.Axios**

Axios is a JavaScript library that we use to make HTTP requests same as fetch( ). There is a difference between these two as in fetch( ), we have to convert the result to a JSON object but in Axios it already returns the result as a JSON object, so we don’t need to convert it.

Call API🡪Set Response in state🡪display data using map.

npm install axios;

import axios from "axios";

**GET:** retrieve data from a server using a specific URL**.**

axios.get("url")

.then((response) => setUser(response.data));

**POST:**

axios.post("https://jsonplaceholder.typicode.com/users",inputData)

.then((response) => console.log (response));

**PUT:**  same as POST

**DELETE:** same as POST