Q.1 Whats React and its pros and cons?

React.js is a free, open-source front-end JavaScript library used for building user interfaces or UI components. It is also known as React or ReactJS.

pros:

Easy to Learn and USe

Creating Dynamic Web Applications Becomes Easier

Reusable Components

Performance Enhancement

The Support of Handy Tools

Known to be SEO Friendly

The Benefit of Having JavaScript Library

Scope for Testing the Codes

cons:

The high pace of development

Poor Documentation

JSX as a barrier

View Part

Q.2 What do you understand by Virtual Dom?

The virtual DOM is the abstraction of the real DOM. In other words, it is the abstraction of an abstraction. A virtual DOM object is the same as a real DOM object, except that it is a lightweight copy. This means that it cannot manipulate on-screen elements. Moreover, upon any change of a property, it only updates the corresponding nodes and not the entire tree. That makes it a quick and efficient alternative.

Q.3 Difference between Virtual Dom vs Real Dom

Virtual Dom:

DOM manipulation is very easy

There is too much memory wastage

No memory wastage

It updates fast

It can't update HTML directly

Update the JSX if the element update

It can produce about 200,000 Virtual DOM

Nodes / Second.

It is only a virtual representation of the DOM

Real Dom:

DOM manipulation is very expensive

There is too much memory wastage

It updates Slow

It can directly update HTML I

Creates a new DOM if the element updates.

It allows us to directly target any specific node (HTML element)

It represents the Ul of your application

Q.4 Whats component? Types of component

A Component is one of the core building blocks of React. In other words, we can say that every application you will develop in React will be made up of pieces called components. Components make the task of building UIs much easier

Types of component:

Functional Components:

Functional components are simply javascript functions. We can create a functional component in React by writing a javascript function. These functions may or may not receive data as parameters, we will discuss this later in the tutorial. The below example shows a valid functional component in React

Class Components:

The class components are a little more complex than the functional components. The functional components are not aware of the other components in your program whereas the class components can work with each other. We can pass data from one class component to another class component. We can use JavaScript ES6 classes to create class-based components in React. The below example shows a valid class-based component in React

Q.5 Difference between class & function based component

class component :

A class component requires you to extend from React. Component and create a render function that returns a React element.

It must have the render() method returning JSX (which is syntactically similar to HTML)

The class component is instantiated and different life cycle method is kept alive and is run and invoked depending on the phase of the class component.

Also known as Stateful components because they implement logic and state.

It requires different syntax inside a class component to implement hooks.

example: constructor(props) {

super(props);

this.state = {name: ‘ ‘}

}

Constructor is used as it needs to store state.

function component :

A functional component is just a plain JavaScript pure function that accepts props as an argument and returns a React element(JSX).

There is no render method used in functional components.

Functional component run from top to bottom and once the function is returned it can't be kept alive.

Also known as Stateless components as they simply accept data and display them in some form, they are mainly responsible for rendering UI.

React lifecycle methods (for example, componentDidMount) cannot be used in functional components.

Hooks can be easily used in functional components to make them Stateful. example: const [name,SetName]= React.useState(' ')

Constructors are not used.

Q.6 Explain react component life cycle

A React Component can go through four stages of its life as follows:

Initialization:

This is the stage where the component is constructed with the given Props and default state. This is done in the constructor of a Component Class.

Mounting:

Mounting is the stage of rendering the JSX returned by the render method itself.

React has four built-in methods that gets called, in this order, when mounting a component:

constructor()

getDerivedStateFromProps()

render()

componentDidMount()

Updating:

Updating is the stage when the state of a component is updated and the application is repainted.

React has five built-in methods that gets called, in this order, when a component is updated:

getDerivedStateFromProps()

shouldComponentUpdate()

render()

getSnapshotBeforeUpdate()

componentDidUpdate()

Unmounting:

As the name suggests Unmounting is the final step of the component lifecycle where the component is removed from the page.

Q.7 Explain Prop Drilling in React & Ways to avoid it

Prop drilling is a situation where data is passed from one component through multiple interdependent components until you get to the component where the data is needed.

Three ways to avoid prop drilling in React:

useContext

component composition

Zustand library