**How does React work?**

React creates a virtual DOM. When state changes in a component it firstly runs a “diffing” algorithm, which identifies what has changed in the virtual DOM. [The second step is reconciliation, where it updates the DOM with the results of diff1](https://www.fullstack.cafe/blog/mern-stack-interview-questions).

**What are React Hooks?**

React Hooks are a feature introduced in React version 16.8 that allow developers to use state and other React features without writing a class component. Hooks provide a way to reuse stateful logic across multiple components.

There are several built-in React Hooks, including:

* useState: This hook allows functional components to have state by providing a way to declare and update state variables. It returns a state variable and a function to update that variable.
* useEffect :In React, the useEffect hook is used to perform side effects in functional components. Side effects can include data fetching, subscriptions, or manually interacting with the DOM. The useEffect hook is called after every render and allows you to manage these side effects. It takes two arguments: a function and an optional array of dependencies.
* useContext:useRef: useMemo: useCallback: (CMR- Short form)

***These are just a few examples of React Hooks. They simplify the development process, improve code readability, and promote the reuse of logic across components.***

Here is an example of using the useState hook to add state to a functional component:

import { useState } from 'react';

function Counter() {

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

return (

<div>

<p>You clicked {count} times</p>

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

Click me

</button>

</div>

);

}

In this example, we use the useState hook to create a count state variable and a setCount function to update it. When the button is clicked, we call setCount with the new count value, which updates the state and causes the component to re-render with the new count value.

**What are props in React?**

Props are inputs to a React component. They are single values or objects containing a set of values that are passed to React Components on creation using a naming convention similar to HTML-tag attributes. They are data passed down from a parent component to a child component.

[The primary purpose of props in React is to provide following component functionality: Pass custom data to your React component, Trigger state changes, Use via this.props.reactProp inside component’s render () method1](https://www.fullstack.cafe/blog/mern-stack-interview-questions).

Here is an example of passing props from a parent component to a child component:

function Greeting(props) {

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

}

function App() {

return <Greeting name="John" />;

}

In this example, the App component passes a name prop with the value "John" to the Greeting component. The Greeting component receives this prop and uses it to render the greeting message.

**What is jsx?**

**"JSX stands for 'JavaScript XML.' It is a syntax extension used with React to write HTML-like code directly within JavaScript. JSX allows developers to define the structure and appearance of React components using a syntax that closely resembles HTML, making it more intuitive and easier to work with when building user interfaces.**

**JSX code is converted by tools like Babel into regular JavaScript code that creates React elements using** React.createElement() **calls.**

**Using JSX has several advantages:**

* **Declarative Syntax: JSX allows us to write code that closely resembles the final UI output, making it easier to reason about the component's behavior.**
* **Readability: The HTML-like syntax of JSX makes the code more readable, especially for developers familiar with HTML/XML.**
* **Component Composition: JSX enables us to compose components using familiar HTML tags, making it simpler to create complex UIs.**
* **Performance: JSX can help optimize rendering performance by efficiently updating the virtual DOM through its diffing algorithm.**

**However, it's essential to note that even though JSX resembles HTML, it's not actually HTML. JSX gets transpiled into JavaScript, and it must adhere to JavaScript rules. For example, class attributes must be written as** className**, and self-closing tags must use a trailing slash (**<br />**) in JSX.**

**Overall, JSX is a powerful tool that enhances the development experience with React, making it one of the main reasons why React has become so popular for building user interfaces."**

1. [**Differentiate between Real DOM and Virtual DOM1**](https://www.edureka.co/blog/interview-questions/react-interview-questions/)**.**

| **DOM manipulation is very expensive** | **DOM manipulation is very easy** |
| --- | --- |
| **There is too much memory wastage** | **No memory wastage** |
| **It updates Slow** | **It updates fast** |
| **It can directly update HTML** | **It can’t update HTML directly** |
| **Creates a new DOM if the element updates.** | **Update the JSX if the element update** |
| **It allows us to directly target any specific**  **node (HTML element)** | **It can produce about 200,000 Virtual DOM**  **Nodes / Second.** |
| **It represents the Ul of your application** | **It is only a virtual representation of the DOM** |

**Q: What is React? A: React is a JavaScript library for building user interfaces. It allows developers to create reusable UI components and manage the application state efficiently. React was developed by Facebook and has gained significant popularity due to its performance, ease of use, and the ability to build complex and interactive web applications.React is often referred to as a "front-end library" or "view library" because it is mainly concerned with rendering the user interface components of a web application.**

**Q: What are the key features of React? List some of the major advantages of React**

**CJOVRR(Chutiya JoVRR)**

* **Components: React allows developers to create reusable and modular UI components, making it easier to manage complex UI structures.**
* **JSX: JSX is a syntax extension used in React that allows developers to write component structures using a mix of JavaScript and HTML-like syntax, making code more readable and expressive.**
* **One-Way Data Binding: React follows a unidirectional data flow, where data flows from parent components to child components, ensuring predictable and maintainable application state management.**
* **Virtual DOM: React uses a Virtual DOM to optimize rendering and minimize direct DOM manipulations, resulting in better performance.**
* **React Hooks: Introduced in React 16.8, hooks allow developers to use state and other React features in functional components, reducing the need for class components.**
* **React Router: React Router is a popular routing library that allows developers to handle client-side routing in React applications.**

**4)what is interpreted programming langage?**

**Give me interview based answer.**

**An interpreted programming language is a type of programming language that uses an interpreter to execute the code.** [**The code in an interpreted programming language does not require a compiler and gets directly executed without compiling into a machine language code1**](https://bing.com/search?q=)**.** [**Python is an example of an interpreted programming language2**](https://www.edureka.co/blog/interview-questions/python-interview-questions/)**.** [**It is widely used by developers and known for its simplicity, readability, and ease of use, making it an ideal choice for beginners2**](https://www.edureka.co/blog/interview-questions/python-interview-questions/)**. Is there anything else you would like to know? 😊**

\*)why js is not multithreaded programming language

[**JavaScript is not a multithreaded programming language because the JavaScript interpreter in the browser is a single thread1**](https://bing.com/search?q=)**.** [**Even Google Chrome will not let a single web page’s JavaScript run concurrently because this would cause massive concurrency issues in existing web pages1**](https://bing.com/search?q=)**.** [**However, you can use setTimeout to allow some sort of scheduling and “fake” concurrency1**](https://bing.com/search?q=)**. This causes the browser to regain control of the rendering thread, and start the JavaScript code supplied to setTimeout after the given number of milliseconds**[**1**](https://bing.com/search?q=)**. Is there anything else you would like to know? 😊**

**Q)What are closers in jsvascript,why or where can we use closures?**

**A closure in JavaScript is a function that has access to variables in its outer scope, even after the outer function has returned. This is because closures are created at function creation time, and they capture the environment in which they were created.**

**Closures can be used to:**

* **Encapsulate data: Closures can be used to encapsulate data so that it is only accessible to the functions that need it. This can be useful for security and for preventing data from being accidentally modified.**
* **Create reusable functions: Closures can be used to create reusable functions that can access data from different scopes. This can be useful for creating functions that can be used in different contexts.**
* **Implement callbacks: Closures can be used to implement callbacks, which are functions that are called later. This can be useful for deferring execution of code until it is needed.**

**Here is an example of a closure in JavaScript:**

**function makeCounter() {**

**var count = 0;**

**function increment() {**

**count++;**

**return count;**

**}**

**return increment;**

**}**

**var counter = makeCounter();**

**console.log(counter()); // 1**

**console.log(counter()); // 2**

**In this example, the makeCounter() function creates a closure that has access to the variable count. The increment() function is a closure that uses this variable to increment the count and return the new value. The counter variable is assigned to the return value of the makeCounter() function. This means that counter is a reference to the closure that was created by makeCounter(). When we call counter(), the closure is executed and the count is incremented.**

**Closures can be used in a variety of places in JavaScript. They are often used in event handlers, in libraries, and in functional programming.**

**Q)difference between functional component and class component**

| **Aspect** | **Functional Component** | **Class Component** |
| --- | --- | --- |
| **Syntax** | **Arrow function with** props **parameter** | **ES6 class syntax with** render **method** |
| **State and Lifecycle** | **Can manage state and lifecycle with hooks** | **Can manage state and lifecycle with built-in methods** |
| **Performance** | **Can be more efficient due to less overhead** | **Can be less efficient due to class instance creation** |
| **Readability** | **Generally simpler and cleaner code** | **More verbose syntax** |
| **Hooks** | **Can use all available hooks** | **No hooks support (React 16.8 and earlier)** |
| **Usage of** this | **No usage of** this**, reducing confusion** | **Uses** this **for accessing props and state** |
|  |  |  |

**Q)diff between tcp and udp, where it is used?**

**what will happen when write google.com and press enter**

**what is interpreted programming langage**

**why js is not multithreaded programming language**

**what are closers in jsvascript,why or where can we use closures**

**advantages of js on the basic js or vanilla js**

**difference between functional component and class component**

**ask weather css is important or not**

**always ask the requirments to the interviewer**

**What are new tags in html5?**

**Q)Inline vs block level elements**

**Q)Can we include inline elements inside block elements**

inline→<span>**,** <a>**,** <strong>**,** <em>**,** <img>**, and more.**

**Block elements→**<div>**: A generic container element used for grouping other elements and applying CSS styles.**

* <p>**: Represents a paragraph of text.**
* <h1>**,** <h2>**,** <h3>**,** <h4>**,** <h5>**,** <h6>**: Headings with different levels of importance.**
* <ul>**: Unordered list, typically used to create lists of items.**
* <ol>**: Ordered list, used for numbered lists.**
* <li>**: List item, used within** <ul> **or** <ol> **to define individual list items.**

**Symmentically incorrect, the product might get break due to framework errors but heml will not give errors...**

**Mutable things are passed by reference such as array, objects when passed by function parameter.**

**Immutable things are passed by value when passed in a function.**

**op of the question is --->4, --->[x,b,c] ------>false**

**\*) 1 4 3 2**

**Q)Event loop**

**Q)How to convert class based component into functional component**

**QHow we can avoid prop drilling?**

**Q)difference between redux and context api**

**Q)What is the use of useReducer?**

**Q)What is middleware ?**

**Q)Why we need a middleware in redux implementation?**

**Q)Why we need to go typescript instead of js?**