1. What is hooks?

Ans: Hooks are functions that let you “hook into” React state and lifecycle features from function components.

1. What is functional component?

Ans : Functional component is just a simple javascript function; it accepts the data in the form of props and returns the react element.

1. Difference between useeffect and usememo?

Ans : The **useEffect** and **useMemo** hooks are two functions that are used to manage state and performance in a [React](https://www.c-sharpcorner.com/topics/react) application.

The **useEffect** hook is used to perform side effects in a React component. This can include things like making a network request, setting up a subscription, or updating the DOM in response to a change in state.

* The **useMemo** hook is used to optimize the performance of a React component by memoizing (caching) the results of a calculation or function. This can be useful in cases where the calculation or function is expensive, and the result is not likely to change often.

1. What is render() ?

Ans : In React, Render is the technique that can redirect a page with the help of function render().

1. Describe state.

Ans : The state is a built-in React object that is used to contain data or information about the component. A component's state can change over time; whenever it changes, the component re-renders.

1. What is api?

Ans : APIs are mechanisms that enable two software components to communicate with each other using a set of definitions and protocols. For example, the weather bureau's software system contains daily weather data. The weather app on your phone “talks” to this system via APIs and shows you daily weather updates on your phone.

1. Describe usereducer with example.

Ans : The useReducer Hook is used to store and update states, just like the useState Hook.

**Exa.,** It accepts a reducer function as its first parameter and the initial state as the second. useReducer returns an array that holds the current state value and a dispatch function to which you can pass an action and later invoke it.

* const [state, dispatch] = useReducer(reducer, initialArg, init);
* An alternative to [useState](https://legacy.reactjs.org/docs/hooks-reference.html" \l "usestate). Accepts a reducer of type (state, action) => newState, and returns the current state paired with a dispatch method.

1. Write a query solution of use context.

Ans : const value = useContext(MyContext);

Createcontext :

Consumer :

Provider :

1. What is component in react?

Ans : Components are independent and reusable bits of code. They serve the same purpose as cJavaScript functions, but work in isolation and return HTML. Components come in two types, Class components and Function components, in this tutorial we will concentrate on Function components.

* Functional component and class component
* Functional component :-

function Welcome(props) {

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

}

* Class component :-
* A class component must include the extends React.Component statement. This statement creates an inheritance to React.Component, and gives your component access to React.Component's functions.
* The component also requires a render() method, this method returns HTML.

Exa., class Car extends React.Component {

render() {

return <h2>Hi, I am a Car!</h2>;

}

} ///////// car call thay 6

1. Write a code of useref.

Ans : input mathi koi data bahar kadhvo hoy tyare useref no use thay

* Useref ni value always null j hoy
* Onclick thi data bahar nikde
* import React, {useRef} from "react";
* const Useref = () => {
* const inputref = useRef(null)
* function inputfunction(){
* console.log("inputfunction");
* inputref.current.value = "100"
* console.log(inputref.current.value);
* inputref.current.focus()
* inputref.current.style.color = "green"
* }
* return (
* <>
* <h1>Useref</h1>
* <input type="text" ref={inputref} />
* <button onClick={inputfunction}>Update</button>
* </>
* );
* }
* export default Useref;

1. use layout

* Ans : The purpose of useLayoutEffect is to let your component use layout information for rendering: Render the initial content. Measure the layout before the browser repaints the screen. Render the final content using the layout information you've read.

1. what is function describe any 4 function in js.

Ans : A function in JavaScript is similar to a procedure—a set of statements that performs a task or calculates a value, but for a procedure to qualify as a function, it should take some input and return an output where there is some obvious relationship between the input and the output.

* **Function declaration :-** The name of the function.
* A list of parameters to the function, enclosed in parentheses and separated by commas.
* The JavaScript statements that define the function, enclosed in curly braces, { /\* … \*/ }.
* **function square(number) {**
* **return number \* number;**
* **}**
* **Function expressions :-** While the function declaration above is syntactically a statement, functions can also be created by a [function expression](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/function).
* const square = function (number) {

return number \* number;

};

console.log(square(4)); // 16

* **Function hoisting :-**

function square(n) {

return n \* n;

}

console.log(square(5)); // 25

* Function hoisting only works with function declarations — not with function expressions.
* **Nested function :-** You may nest a function within another function. The nested (inner) function is private to its containing (outer) function.
* The inner function can be accessed only from statements in the outer function.
* The inner function forms a closure: the inner function can use the arguments and variables of the outer function, while the outer function cannot use the arguments and variables of the inner function.

function addSquares(a, b) {

function square(x) {

return x \* x;

}

return square(a) + square(b);

}

console.log(addSquares(2, 3)); // 13

console.log(addSquares(3, 4)); // 25

console.log(addSquares(4, 5)); // 41

* **Arrow function :-** An [arrow function expression](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Functions/Arrow_functions) (also called a fat arrow to distinguish from a hypothetical -> syntax in future JavaScript) has a shorter syntax compared to function expressions and does not have its own [this](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/this), [arguments](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Functions/arguments), [super](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/super), or [new.target](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/new.target). Arrow functions are always anonymous.
* An arrow function does not have its own this; the this value of the enclosing execution context is used. Thus, in the following code, the this within the function that is passed to setInterval has the same value as this in the enclosing function:
* function Person() {

this.age = 0;

setInterval(() => {

this.age++; // `this` properly refers to the person object

}, 1000);

}

const p = new Person();

1. What is spa and describe?

Ans : React's fast virtual DOM and component-based design, developers can easily build sophisticated and responsive web apps.

* SPA : - Single page application
* Fast and responsive user experience
* Better offline functionality
* Simplified code maintenance and updates //// Advantages

1. What is use callback?

Ans : useCallback is a hook that will return a memoized version of the callback function that only changes if one of the dependencies has changed.

1. Definitions of all hooks.

Ans : -> **Usestate :-** useState is React Hook that allows you to add state to a functional component.

* **Useeffect :-** The useEffect Hook allows you to perform side effects in your components. Some examples of side effects are: fetching data, directly updating the DOM, and timers. useEffect accepts two arguments. The second argument is optional.
* **Uselayout :-** The purpose of useLayoutEffect is to let your component use layout information for rendering: Render the initial content. Measure the layout before the browser repaints the screen. Render the final content using the layout information you've read.
* **Usememo:-** useMemo is a React Hook that lets you cache the result of a calculation between re-renders. const cachedValue = useMemo(calculateValue, dependencies)
* **Usecallback :-** useCallback is a hook that will return a memoized version of the callback function that only changes if one of the dependencies has changed.
* **Useref :-** useCallback is a hook that will return a memoized version of the callback function that only changes if one of the dependencies has changed.
* **Usecontext :-** The useContext Hook provides function components access to the context value for a context object. It: Takes the context object (i.e., value returned from React. createContext ) as the one argument it accepts. And returns the current context value as given by the nearest context provider.
* **Usecontexthook :-** The useContext Hook provides function components access to the context value for a context object. It: Takes the context object (i.e., value returned from React. createContext ) as the one argument it accepts. And returns the current context value as given by the nearest context provider.
* **Usenavigate :-** The useNavigate hook is part of the react-router-dom package that allows programmatic routing inside a React application.