

Question 1

Tell us differences between uncontrolled and controlled components.

Controlled and uncontrolled components are terms used to describe React components that render HTML form elements. These components are used to handle user input and manage their value and state.

A **controlled component** is one that takes its current value through props and notifies changes through callback functions. It manages its own state and passes the new values as props to the controlled component.

A **uncontrolled component** is one that stores its own state internally, and we query the DOM using a ref to find the HTML.

Question 2

How to validate React props using PropTypes ?

React props are attributes that you can pass from one component to another. They are read-only and can be of different types like strings, numbers, arrays, objects, etc. To validate React props using PropTypes, we need to do the following steps:

Install the prop-types library using npm or yarn. This library provides a range of validators for checking the type of props.

Import PropTypes from "prop-types" in your component file.

Define the propTypes property on our component and assign it an object with the prop names and validators as keys.

Optionally, you can also define the defaultProps property on your component and assign it an object with the prop names and default values as keys.

Question 3

Tell us the difference between nodejs and expressjs.

Node.js and **Express.js** are not comparable in terms of functionality, because they are different in terms of abstraction.

Node.js is a JavaScript runtime built on Chrome's V8 JavaScript engine. It is used to run JavaScript code outside of a browser.

Node.js provides a low-level API for working with HTTP requests, responses, streams, buffers, events, etc. **Node.js** is a platform that allows developers to install and use thousands of third-party packages.

Express.js is a web framework based on **Node.js** that simplifies the development of web applications using **Node.js**. It provides features like templating, error handling, static file serving, etc. that make it easier to organize and structure your code and handle common tasks. Express.js is one of the most popular frameworks in the **Node.js** ecosystem and has many extensions and plugins available.

In summary, **Node.js** is a platform for running JavaScript code on servers, while **Express.js** is a framework for building web applications. We can use **Node.js** without **Express.js**, but we cannot use **Express.js** without **Node.js**.

Question 4

What is a custom hook, and why will we create a custom hook?

A **custom hook** is a function that starts with the word "use" and can call other hooks inside it. **Custom hooks** allow us to reuse logic across multiple components without changing their structure.

Custom hooks can return anything we want, such as values, functions, or effects.

We might create a **custom hook** when we want to extract some common logic from our components and share it across them, such as fetching data from an API, for accessing the local storage, for subscribing to an event, or for validating a form.