

Question answer

Tell us the differences between uncontrolled and controlled components.

Answer Lorem ipsum the main difference between controlled and uncontrolled components in React is that controlled components manage their own state internally using DOM APIs. Controlled components are more flexible and robust, while uncontrolled components rely on the browser's default behavior.

How to validate React props using PropTypes

Answer PropTypes is a built-in library in React that is used to validate the type and shape of the props passed to a React component. It helps catch errors early in development and ensures that components receive the right props.

To use PropTypes, you need to import it from the 'prop-types' module and then define the propTypes object in your component.

Tell us the difference between nodejs and express js.

Answer Node.js and Express.js are both important technologies for building server-side web applications in JavaScript. However, they serve different purposes.

Node.js is a JavaScript runtime environment for executing JavaScript code on the server-side. It provides a platform for building scalable network applications, handling file system operations, and database interactions. Node.js comes with a built-in set of libraries and APIs that allow developers to build server-side applications.

Express.js, on the other hand, is a web application framework built on top of Node.js. It provides a number of features and tools to simplify the process of building web applications, such as routing, handling requests, and managing middleware. While Node.js provides a lot of functionality out of the box, Express.js makes it easier to build web applications.

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

Answer A custom hook is a function in React that encapsulates reusable logic or behavior that can be shared across multiple components. It helps to organize code and avoid repetition throughout their application.

Custom hooks are built using the built-in hooks provided by React, such as useState, useEffect, useContext, and more. By encapsulating logic into custom hooks, developers can create reusable and modular code that is easy to maintain and reuse.

There are several reasons why you might create a custom hook. For example, you might create a custom hook to encapsulate a specific piece of logic or behavior that is used in multiple components. Custom hooks can also help to promote code reuse, reduce code duplication, and improve the overall maintainability of your application.