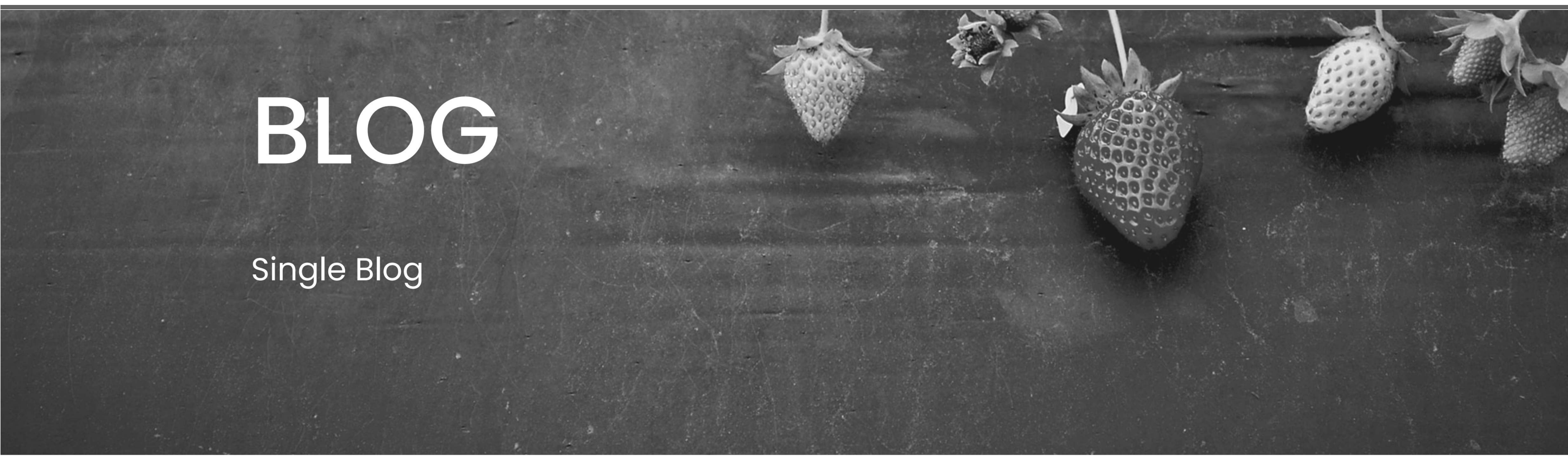

The differences between controlled and uncontrolled components

Controlled Component: A controlled component is a form input element whose value is controlled by React state. In other words, React controls and manages the value of the input field.

Uncontrolled Component: An uncontrolled component is a form input element that manages its own state internally, without relying on React state.

#	CONTROLLED COMPONENT	UNCONTROLLED COMPONENT
1	Does not maintain its internal state.	Maintains its internal state.
2	Data is controlled by the parent component.	Data is controlled by the DOM itself.
3	Accepts its current value as a prop.	Uses a ref for their current values.
4	Allows validation control.	Does not allow validation control.
5	Better control over the form elements and data.	Limited control over the form elements and data.

How to validate React props using PropTypes?

might require a prop to be defined, otherwise, it will not render properly. For using PropTypes we need to import it first.

The PropTypes utility exports a wide range of validators for configuring type definitions like,

- `PropTypes.any:` The prop can be of any data type
- `PropTypes.bool:` The prop should be a Boolean
- `PropTypes.number:` The prop should be a number
- `PropTypes.string:` The prop should be a string
- `PropTypes.func:` The prop should be a function
- `PropTypes.array:` The prop should be an array
- `PropTypes.object:` The prop should be an object
- `PropTypes.symbol:` The prop should be a symbol

Use the component: Now, whenever you use MyComponent, React will perform the prop type validation and provide warnings in the console if the passed props don't match the specified types.

The difference between nodejs and express js

Node JS: Node.js is an open source and cross-platform runtime environment for executing JavaScript code outside of a browser. NodeJS is not a framework and it's not a programming language. It is used for building back-end services like APIs like Web App or Mobile App.

Express JS: Express is a small framework that sits on top of Node.js's web server functionality to simplify its APIs and add helpful new features. It makes it easier to organize your application's functionality with middle ware and routing. It adds helpful utilities to Node.js's HTTP objects. It facilitates the rendering of dynamic HTTP objects.

#	Node JS	Express JS
Usage	It is used to build server-side, input-output, event-driven apps.	It is used to build web-apps using approaches and principles of Node.js.

Block	Google’s V8 engine.	
Level of features	Fewer features.	More features than Node.js.
Written in	C, C++, JavaScript	JavaScript
Controllers	Controllers are not provided.	Controllers are provided.
Routing	Routing is not provided.	Routing is provided.
Middleware	Doesn’t use such a provision.	Uses middleware for the arrangement of functions systematically server-side.

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

A custom hook is a JavaScript function in React that allows you to reuse stateful logic across multiple components. It's a way to extract common functionality into a reusable hook that can be used in different parts of your application.

Custom hooks follow a specific naming convention by starting with the word "use" (e.g., useCustomHook). By using custom hooks, you can encapsulate complex logic, manage state, perform side effects, and share code between different components without the need for duplication.

There is also some reasons for using custom hook such as Reusability, Separation of Concerns, Code Organization, Sharing Stateful Logic, Testing etc.

Overall, custom hooks offer a powerful way to extract, reuse, and share logic in React applications, promoting code reusability, maintainability, and improved development efficiency.