what is the difference between storing data in a variable and storing it in a state?

storing data in a variable:

function Example1() {

const a = 1

return <div>{ a }</div>

}

storing it in a state:

In React, whenever we are working with any data, we always use state for storing that data which may be a string, number or any complex object.

This is fine if you are using that state while rerendering the component or If you want to do something when the state value changes but If you are using that state just for storing data and not using it for rendering or not passed as prop to other components then you should not use state.

Because whenever the state value changes, React will re-render the component and also all its child components will get re-rendered.

function Example2() {

const [a] = useState(1)

return <div>{ a }</div>

}

what is the difference between if statement and ternary operator in conditional rendering?

If statement:

render() {

const isLoggedIn = this.state.isLoggedIn;

let button;

if (isLoggedIn) {

button = <LogoutButton onClick={this.handleLogoutClick} />;

} else {

button = <LoginButton onClick={this.handleLoginClick} />;

}

return (

<div>

<Greeting isLoggedIn={isLoggedIn} />

{button}

</div>

);

}

}

Ternary operator:

In React, there are different ways to conditionally render content based on the state of a component or other conditions. Two common ways are using the ternary operator and the && operator.

function Greeting(props) {

const isLoggedIn = props.isLoggedIn;

return (

<div>

{isLoggedIn ? (

<h1>Welcome back!</h1>

) : (

<h1>Please sign up.</h1>

)}

</div>

);

}

what is the difference between controlled form and uncontrolled form?

In a controlled component, form data is handled by a React component. The alternative is uncontrolled components, where form data is handled by the DOM itself. For example, this code accepts a single name in an uncontrolled component:

class NameForm extends React.Component {

constructor(props) {

super(props);

this.handleSubmit = this.handleSubmit.bind(this);

this.input = React.createRef();

}

handleSubmit(event) {

alert('A name was submitted: ' + this.input.current.value);

event.preventDefault();

}

render() {

return (

<form onSubmit={this.handleSubmit}>

<label>

Name:

<input type="text" ref={this.input} />

</label>

<input type="submit" value="Submit" />

</form>

);

}

}

In HTML, form elements such as <input>, <textarea>, and <select> typically maintain their own state and update it based on user input. In React, mutable state is typically kept in the state property of components, and only updated with [setState()](https://legacy.reactjs.org/docs/react-component.html" \l "setstate).

We can combine the two by making the React state be the “single source of truth”. Then the React component that renders a form also controls what happens in that form on subsequent user input. An input form element whose value is controlled by React in this way is called a “controlled component”.