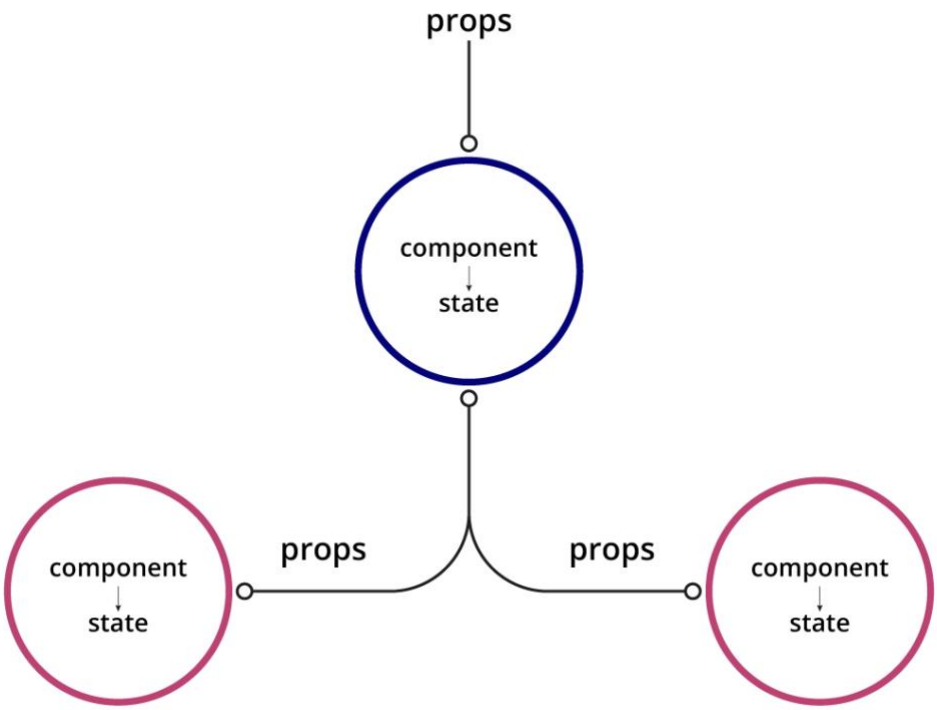
SHOULD I USE PROPS OR STATE?

Use state to store the data your current page needs in your controller-view. Use props to pass data & event handlers down to your child components. These lists should help guide you when working with data in your components.

What is difference between state and props?

The difference comes in which component the data are owned. **State is owned locally and the component itself updates it. Props are owned and read-only by a parent**. Props can be changed only if an upstream shift is caused by a callback function passed on to the child.
A major confusion for frontend developers when they are first learning react, is understanding the difference between state and props. React has the ability to manage the data and properly render the application when the data changes. There are two main ways to think about data, that is props and states. In this article, we will explore the differences between the react state vs props and their purpose. React state and props both provide the details about the item but have differences. Let's take a look at how to pass props into the component and how to update a component's state.



What is State?

The status of a react class component instance can be described as an object of a collection of observed features that control the component's behavior. In other words, the component's state is an entity with some details that can alter during the component's lifetime.

This data is kept inside a component. This particular aspect is local or owned. The component itself uses the setStatus function to update the state.

The state is an updated structure that is used to contain and can modify data or information about the component over time. The state change may occur as a user activity or device event response. It decides the actions of the component and how it is to be done by the core of the React component. As easy as possible, a state should be held. It is the local state or details of the part. Only in the component or by the component directly can it be accessed or changed. Components of react have a built-in state object. The state object is where the property values belonging to the item are stored. The component is restored when the state object changes.

What are Props?

React is a library focused on components that separate the user interface into reusable little bits. These components have to transmit (send data to each other) in some cases, and the way to transmit data among components is through props."Props" is a special React keyword for proprietary purposes, used for data transmission from component to component. But the key part of this is the transmission of data with props in a uniform flow. (parent-to-child route).In addition, props data are read-only, meaning that parent data cannot be modified by child elements.

State vs Props [Comparison Table]

The below comparison table will explain the differences in state vs props.

State	Props

State is managed within the component	**Props gets passed to the component**
State can be changed(mutable)	**Props are read only and cannot be changed (immutable)**
State can be accessed using the use state hooks in functional components and in-class components can be accessed using this. State	**Props can be accessed in functional component using props parameter and in-class component, props can the accessed using this.props**
State changes can be asynchronous	**Props are read only**
State is controlled by react components	**Props are controlled by whoever renders the components**
State can used to display changes with the component	**Props are used to communicate between components**

Difference Between State and Props in React.js

React is an open-source JavaScript library that offers a visual overview of the JavaScript architecture for conventional MVC. React promises programmers a model in which substrates cannot directly influence enclosing components—data are downstream, data changes in HTML are efficiently modified, and the DOM is abstracted to boost performance using Virtual DOM.How can we access data from the previous component if the data is just being flowed from component to component? The answer is props. React uses data transmission props that we need to transfer to various components. The difference comes in which component the data are owned. The state is owned locally, and the component itself updates it. Props are owned and read only by a parent. Props can be changed only if an upstream shift is caused by a callback function passed on to the child.A prop can be passed on to the child in the state of a parent. They apply to the same value but can only be updated by the parent variable.

How are Props Passed into the Component?

We can pass props to any component by declaring HTML tag attributes.

```
<DemoComponent sampleProp = "HelloProp" />
```

We transfer a sampleProp in the above code snippet to the DemoComponent part. This prop has the ‘HelloProp’ value. See how we can access this advice now.In the class components to which the props are transferred, we can access any props.

this.props.propName;

We can use the above syntax to access any prop from within a component class. The “this.props” is a type of total object that stores all props from an item. The propname, which is the propName, is the key.

Passing information from one component to other:

This is one of React’s coolest characteristics. We should make it possible for components to communicate. To understand this, we will look at two components Parent and Child. We will pass information to the Child component as advice from our parent component. We can offer a part as much advice as we want. The content of a prop is not permitted to be changed. No matter what kind of component it is, nobody is allowed to change their advice, whether functional or class-based. The difference comes in which component the data are owned. State is owned locally and the component itself updates it. Props are owned and read-only by a parent. Props can be changed only if an upstream shift is caused by a callback function passed on to the child. A prop can be passed on to the child in the state of a parent. They apply to the same value but can only be updated by the parent variable.

How Do You Update a Component’s State?

Although a react component can have an initial state, the actual power is in updating the state — the component should not be in either state if we don’t have to update the state. State is reserved only for data that changes our part and can be seen in the user interface.We use this.setState() instead of changing the state directly using this.state (). This is a feature for all components that use state, and allows us to inform React that the state of the component has changed. This way the component knows that it can return because its status has changed and the user interface will probably change as well. It is very efficient to use a setter function like that.

React intentionally waits until all components in their event handlers call setState() before they start returning. This increases efficiency by preventing excessive re-renders.

You may also ask why React does not update this, however.

Two major reasons exist:

- The consistency of props and the state is broken, which causes problems that are very difficult to debug.
- This will make it difficult to introduce such new features.
- React will load several setState() calls for performance into a single update.
- Due to the asynchronous of this.props and this.state, you cannot depend on their values for the next state to be calculated.

To fix it, use a second setState() form, which accepts a function instead of an object. This function is the first argument for the previous state, and the props are the second argument when the update is applied:

```
this.setState(function(state, props) {

  return {

    counter: state.counter + props.increment

  };

});
```

Is State Changeable?

A state change takes place on the basis of the user input, which triggers an occurrence. React (with status) components are often made on the basis of state data. The initial knowledge is held by the State.Thus when the state changes, React will be notified and the DOM will be re-rendered immediately; not the whole DOM but only the modified portion. This is one of the reasons for the fast reaction.

And how do you notify React? You thought: with setState(). The setState() method triggers the mechanism for rendering the modified components. React is notified, knows which part(s) to alter, and does so quickly without restoring the entire DOM.

Is State Created in the Component?

Let's see the constructor method:

```
constructor() {

  super();

  this.state = {

    count: 0,

  };

}
```

This is where the state gets the initial data. The initial data (as above) can be hard-coded, but it can also come from props.However, it makes sense – you cannot adjust props but the data a component receives wants to do so. This is where the state enters.

Component types

- **Stateless component:** Just props, no state. Besides the render() function, there's not much going on and all its logic is about the props that they get. This makes it easy to track them (and test for that matter).
- **The stately component:** state as well as props. These are also called state managers. They are responsible for communication between clients and their servers (XHR, Web sockets, etc.), data processing, and user events.

What Happens When State Changes?

React Components allow you to break the UI into separate, reusable components so that you can look into every single item on an isolated basis.Components are conceptually like functions in JavaScript. They accept arbitrary inputs and return elements of react that describe what should be shown on the screen. If you have to allow the user to enter something or to alter the variables that the component is supported by, you would have to setState.State allows React components in response to user behavior, network responses, and everything else to adjust their performance over time, without violating this rule. Class-defined components provide additional functionality. Local status is the only class component function available.

Can I Use State in Every Component?

In the early days, only class components, and not functional components, were included.That's why stateless components are known for their functional components. However, state can now be used in both class and functional components following the implementation of React Hooks.

You can only use status in class components if your project does not use React Hooks.

React State Examples

Every react component returns JSX which describes the user interface and state is one of ways through which you can influence what is rendered on the screen as the User interface.JSX is an extension of javascript to write XML-like code for elements and components. There are two ways to initialize the state in react component.

1. Inside the constructor
2. Directly inside the class
3.

State or Props: Which to Use and When?

Every time anything is changed or modified in the application, think of using state, and state can be passed to components through props. props in simple terms are the way components communicate to one another and we can pass props through react using different attributes and access these attributes in other components using **this.props.name**. The benefit of this is your state might have to access the same state in different places via your application and you can do this via props.

The Component State and Props: Similarities

Props and states both provide details about the item, but they are used differently and must be kept separate.

Conclusion

State refers to the component's local status which cannot be used and changed outside the component and can only be used and modified within the component.

On the other hand, it provides reusable components by enabling components to obtain data in the form of props from the parent component. We may change the state of a component with setState. These notifications are also triggered by events. setState is called asynchronous and merged with every entity in the current state. We may also transfer a setState function to allow us to write status changes based on the current status values. Most of your components can be stateless when you create an app. Props transfer parent-to-child data. They are unchangeable and are thus unchanged.