**Programming in React JS**

* What are Props

Props stand for "Properties." They are read-only components. It is an object which stores the value of attributes of a tag and work similar to the HTML attributes. It gives a way to pass data from one component to other components. It is similar to function arguments. Props are passed to the component in the same way as arguments passed in a function.

Props are immutable so we cannot modify the props from inside the component. Inside the components, we can add attributes called props. These attributes are available in the component as this.props and can be used to render dynamic data in our render method.

When you need immutable data in the component, you have to add props to reactDom.render() method in the main.js file of your ReactJS project and used it inside the component in which you need. It can be explained in the below example.

* Lets Practice Props

**Example**

**App.js**

import React, { Component } from 'react';

class App extends React.Component {

   render() {

      return (

          <div>

            <h1> Welcome to { this.props.name } </h1>

            <p> <h4> This is the example of props. </h4> </p>

          </div>

      );

   }

}

export default App;

**Main.js**

import React from 'react';

import ReactDOM from 'react-dom';

import App from './App.js';

ReactDOM.render(<App name = "PropsExample" />, document.getElementById('app'));

**Output:**

Welcome to PropsExample

This is the example of props.

* What is State in React JS

The state is a built-in React object that is used to contain data or information about the [component.](https://www.simplilearn.com/tutorials/reactjs-tutorial/reactjs-components) A component’s state can change over time; whenever it changes, the component re-renders. The change in state can happen as a response to user action or system-generated events and these changes determine the behavior of the component and how it will render.

|  |
| --- |
| class Greetings extends React.Component {    state = {      name: "World"    };    updateName() {      this.setState({ name: "Simplilearn" });    }    render() {        return(            <div>                {this.state.name}            </div>        )    }  } |

* A state can be modified based on user action or network changes
* Every time the state of an object changes, React re-renders the component to the browser
* The state object is initialized in the constructor
* The state object can store multiple properties
* this.setState() is used to change the value of the state object
* setState() function performs a shallow merge between the new and the previous state
* ReactJS(Diff Props vs State)

|  |  |
| --- | --- |
| PROPS | STATE |
| The Data is passed from one component to another. | The Data is passed within the component only. |
| It is Immutable (cannot be modified). | It is Mutable ( can be modified). |
| Props can be used with state and functional components. | State can be used only with the state components/class component (Before 16.0). |
| Props are read-only. | State is both read and write. |

* Why Props are Read-only

Whether you declare a component [as a function or a class](https://reactjs.org/docs/components-and-props.html#function-and-class-components), it must never modify its own props. Consider this sum function:

function sum(a, b) {

return a + b;

}

Such functions are called [“pure”](https://en.wikipedia.org/wiki/Pure_function) because they do not attempt to change their inputs, and always return the same result for the same inputs.

In contrast, this function is impure because it changes its own input:

function withdraw(account, amount) {

account.total -= amount;

}

* Composition of React Components

Component composition is the name for passing components as props to other components, thus creating new components with other components.

**Example:**

const Button = ({ onClick, children }) => (

<button onClick={onClick}>{children}</button>

);

const App = () => {

const onClick = () => alert('Hey');

return (

<Button onClick={onClick}>Click me!</Button>

);

};

* Stateless Component

Stateless components are those components which don’t have any state at all, which means you can’t use this.setState inside these components. It is like a normal function with no render method. It has no lifecycle, so it is not possible to use lifecycle methods such as componentDidMount and other hooks. When react renders our stateless component, all that it needs to do is just call the stateless component and pass down the props.

import React from 'react';

function Example(props) {

return(

<div>

<p>{props.first\_name}</p>

<p>{props.last\_name}</p>

</div>

)

}

export default Example;

* Stateful Component

Stateful components are those [components](https://www.cronj.com/blog/difference-container-component-react-js/) which have a state. The state gets initialized in the constructor. It stores information about the component’s state change in memory. It may get changed depending upon the action of the component or child components.

import React, { Component } from 'react';

class StateExample extends Component {

constructor(){

super();

this.state={

first\_name: 'Shruti',

last\_name: 'Priya'

}

}

render(){

return (

<div>

<p> Class Component </p>

<p>{this.state.first\_name}</p>

<p>{this.state.last\_name}</p>

</div>

)

}

}

export default StateExample;

* When would you use a stateless Component
* When you just need to present the props
* When you don’t need a state, or any internal variables
* When creating element does not need to be interactive
* When you want reusable code
* When would you use a stateful Component
* When building element that accepts user input
* ..or element that is interactive on page
* When dependent on state for rendering, such as, fetching data before rendering
* When dependent on any data that cannot be passed down as props
* Stateful Component vs Stateless Component

|  |  |
| --- | --- |
| Stateful Component | Stateless Component |
| a stateful component has a state object. | a stateless components doesn’t. |
| stateful component owns it’s own state object and is dependent on it’s properties | a stateless component only receives a props object |
| A stateless component is usually associated with how a concept is presented to the user. | A stateful component is always a class component. It is created by extending the React.Component class. A stateful component is dependent on it’s state object and can change it’s own state. |
| These are also know as Class based or Container or Smart Components. | These are also known as Functional based or Presentational or Dumb Components. |
| These have a state object. | These don’t have a state object. |
| These keep a track of changing data via the state object. | These print out what is given to them via props, or they always render the same thing. |