React 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.

Example

App.js

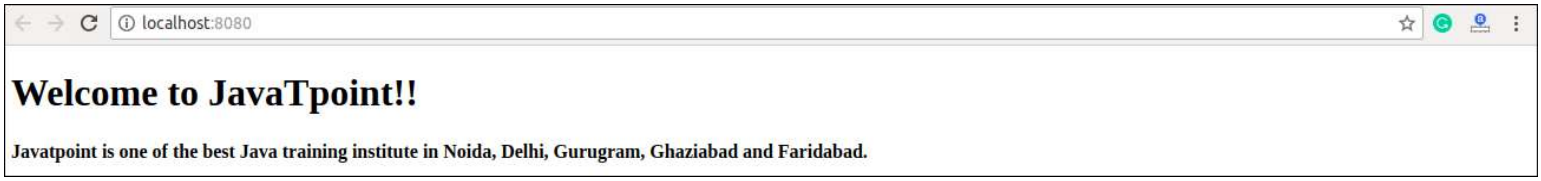
```
import React, { Component } from 'react';
class App extends React.Component {
  render() {
    return (
       <div>
         <h1> Welcome to { this.props.name } </h1>
         <p> <h4> Javatpoint is one of the best Java training institute in Noida, Delhi, Gurugram, Ghaziabad
       </div>
    );
  }
}
export default App;
```

Main.js

```
import React from 'react';
import ReactDOM from 'react-dom';
import App from './App.js';
```

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

Output



Default Props

It is not necessary to always add props in the reactDom.render() element. You can also set **default** props directly on the component constructor. It can be explained in the below example.

Example

App.js

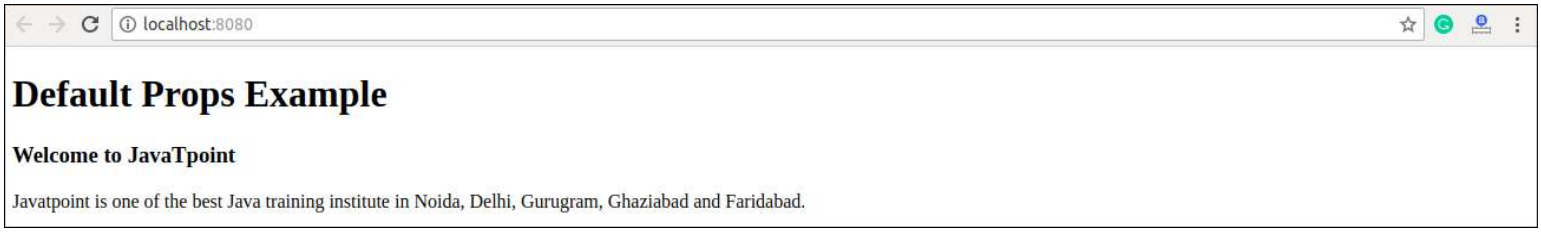
```
import React, { Component } from 'react';
class App extends React.Component {
  render() {
    return (
        <div>
          <h1>Default Props Example</h1>
         <h3>Welcome to {this.props.name}</h3>
         <p>Javatpoint is one of the best Java training institute in Noida, Delhi, Gurugram, Ghaziabad and Fa
</p>
        </div>
      );
   }
}
App.defaultProps = {
  name: "JavaTpoint"
}
export default App;
```

Main.js

```
import React from 'react';
import ReactDOM from 'react-dom';
import App from './App.js';

ReactDOM.render(<App/>, document.getElementById('app'));
```

Output



State and Props

It is possible to combine both state and props in your app. You can set the state in the parent component and pass it in the child component using props. It can be shown in the below example.

Example

App.js

```
import React, { Component } from 'react';
class App extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      name: "JavaTpoint",
    }
  }
  render() {
    return (
      <div>
        <JTP jtpProp = {this.state.name}/>
      </div>
    );
```

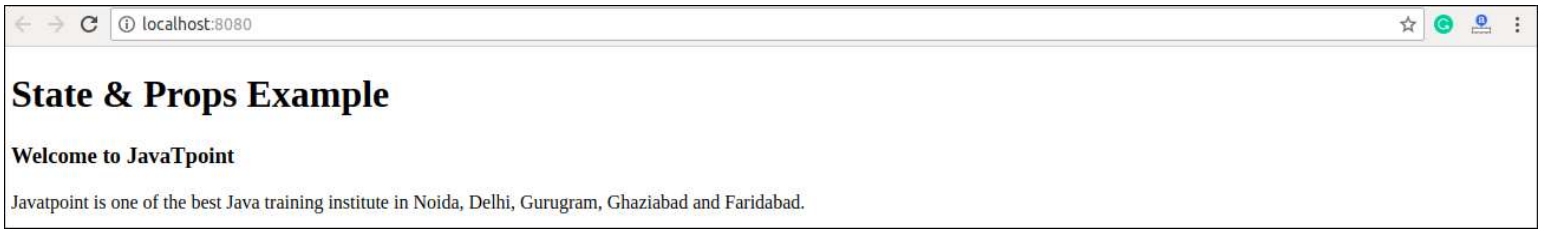
```
  }
}
class JTP extends React.Component {
  render() {
    return (
        <div>
          <h1>State & Props Example</h1>
          <h3>Welcome to {this.props.jtpProp}</h3>
          <p>Javatpoint is one of the best Java training institute in Noida, Delhi, Gurugram, Ghaziabad and F
</p>
        </div>
    );
  }
}
export default App;
```

Main.js

```
import React from 'react';
import ReactDOM from 'react-dom';
import App from './App.js';

ReactDOM.render(<App/>, document.getElementById('app'));
```

Output:



← Prev

Next →