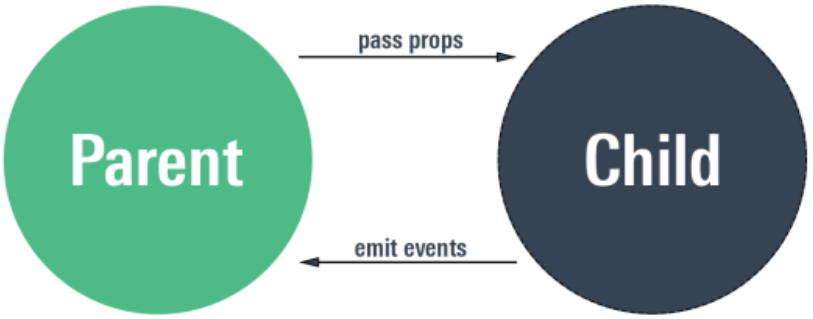
Props In React.Js



Props stand for "**Properties**." They are **read-only** components. It is an object which stores the value of **attributes** of a tag and works similarly 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.

• Passing and Accessing props

We can pass props to any component as we declare attributes for any HTML tag. Have a look at the below code snippet:

<DemoComponent sampleProp = "HelloProp" />

In the above code snippet, we are passing a **prop** named **sampleProp** to the component called **DemoComponent**. This prop has the value "**HelloProp**". Let us now see how we can access these props.
We can access any props inside from the component's class to which the props are passed. The props can be accessed as shown below:

this.props.propName;

We can access **any** prop from inside a **component's** class using the above **syntax**. The '**this.props**' is a kind of global object which stores all of a component's props. The **propName**, that is the names of props are keys to this object.
Below is a sample program to illustrate how to pass and access **props** from a **component**:

Open your react project and edit the **App.js** file in the src folder:
src App.js:

```
import React from 'react';
import ReactDOM from 'react-dom;

// sample component to illustrate props
class DemoComponent extends React.Component{
   render(){
      return(

            <div>
               {/*accessing information from props */}
               <h2>Hello {this.props.user}</h2>
               <h3>Welcome to Coding Ninjas</h3>
            </div>
         );
   }
}

ReactDOM.render(
   // passing props
   <DemoComponent user = "Ayush Goel" />,
   document.getElementById("root")
);
```

*In the above example, we have used a **class-based** component to illustrate **props**. But we can pass props to a **function-based** component also just like we did in the above example. But to access a prop from a function we do not need to use the **'this'** keyword anymore.*

Passing information from one component to another:

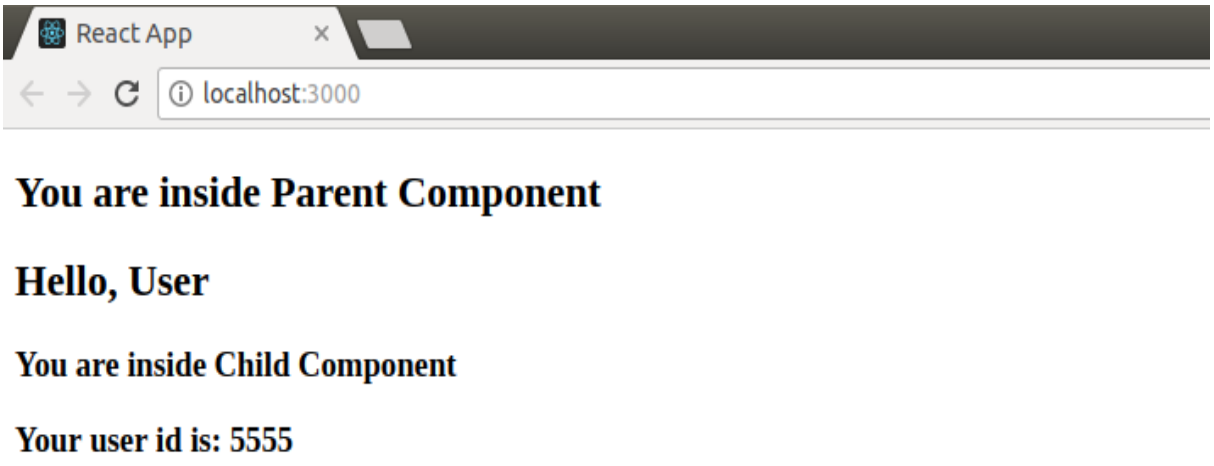
This is one of the **coolest** features of React. We can make **components** to **interact among themselves**. We will consider two components, **Parent** and **Children,** to explain this. We will pass some information as props from our **Parent component to the Child component.** We can pass as many props as we want to a component.
Look at the below code:

Open your react project and edit the **App.js** file in the **src** folder:
src **App.js**:

```
import React from 'react';
import ReactDOM from 'react-dom;

// Parent Component
class Parent extends React.Component{
        render(){
                return(
                                <div>
                                        <h2>You are inside Parent Component</h2>
                                        <Child name="User" userId = "5555"/>
                                </div>
                        );
        }
}

// Child Component
class Child extends React.Component{
        render(){
```

```
            return(
                        <div>
                            <h2>Hello, {this.props.name}</h2>
                            <h3>You are inside Child Component</h3>
                            <h3>Your user id is: {this.props.userId}</h3>
                        </div>
                    );
        }
}

ReactDOM.render(
        // passing props
        <Parent />,
        document.getElementById("root")
);
```

Output:



So we have seen **props** in **React** and also have learned about how props are used, how they can be passed to a **component**, how they are accessed inside a component, and much more. We have used the thing "**this.props.propName**" very often in this complete scenario to access props.