Programming in ReactJS

* Passing props to child components

**Steps:**

* Embed the child component to the parent component.
* Pass the data variable (props) with an assigned value to the child component as an argument while embedding it to the parent component.
* If a user wants to pass multiple data variables (props), all variable names should be unique.
* In the child component, access the data variable value using ‘this.props.variable\_name’.

**Example:**

**Filename: App.js** In this file, we will embed the child component to the parent component. Also, we pass multiple props to the child component.

|  |
| --- |
| import React, { Component } from 'react';  import Child from './components/child.js';  // Child component embedded to parent component  // with props value  class App extends Component {    render() {      return (        <div>          <h1>This is a parent component</h1>          <ul>           <li>            <Child name={"alex"} age={10}/>           </li>          </ul>        </div>      );    }  }  export default App; |

**Filename: Child.js** In the child.js file, we will access props from the parent component and render them inside the child component.

|  |
| --- |
| import React, { Component } from 'react';  // Accessing the props value using this.props  class Child extends Component {    render() {      return (        <div>          <h1>This is a child component</h1>          <h1>My Name is  { this.props.name }          </h1>          <h1>I am  { this.props.age }years old.          </h1>        </div>      );    }  }  export default Child; |

 Output:

This is a child component

My Name is alex

I am 10 years old.

* Updating state of React JS component
* **Steps:**
* Go inside the App.js file and clear everything.
* At the top of the App.js file import React,{Component} from ‘react‘.
* Create a Class based component named ‘App’. This is the default App component that we have reconstructed.
* Create a state object named text, using this.state syntax. Give it a value.
* Create another method inside the class and update the state of the component using ‘this.setState()’ method.
* Pass the state object in a JSX element and call the method to update the state on a specific event like button click.

// The App.js file

import React,{Component} from 'react';

class App extends Component {

constructor(){

super()

this.state={

text : ‘Welcome to ReactJs’

}

}

goPremium(){

this.setState({

text:'Subscription successful'

})

}

render() {

return (

<div><h1>{this.state.text}</h1>

<button onClick={() => this.goPremium()}>Go Premium

</button>

</div>

);

}

}

export default App;

Output:

Before

Welcome to ReactJs

Go Premium

After clicking button

Subscription successful

Go Premium

* What is the significant of SetState function

To change a value in the state object, use the this.setState() method.

When a value in the state object changes, the component will re-render, meaning that the output will change according to the new value(s).

**Example:**

Add a button with an onClick event that will change the color property:

class Car extends React.Component {

constructor(props) {

super(props);

this.state = {

brand: "Ford",

model: "Mustang",

color: "red",

year: 1964

};

}

changeColor = () => {

this.setState({color: "blue"});

}

render() {

return (

<div>

<h1>My {this.state.brand}</h1>

<p>

It is a {this.state.color}

{this.state.model}

from {this.state.year}.

</p>

<button

type="button"

onClick={this.changeColor}

>Change color</button>

</div>

);

}

}

**Output:**

Before

# My Ford

It is a red Mustang from 1964.

Change Color

After

# My Ford

It is a blue Mustang from 1964.

Change Color

Always use the setState() method to change the state object, it will ensure that the component knows its been updated and calls the render() method (and all the other lifecycle methods).