REACT-Components Communication Using Props & State

Components Communication

We are going to cover the following topics for communication:

- What is Props?
- Using Props with Class Components
- Using Props with Functional Components
- What is State?
- State Vs Props

What is Props?

- Props are the short name for properties and are immutable.
- Any information that gets passed from one component to another is known as ‘props’.
- Props are passed to components via HTML attributes.

- The component receives the argument as a props object.
- To access that information we have to use the JSX expression **this.props**.

Using Props with Class Components

Create 2 (Class) Components: One as a Parent and another one as a Child.

Parent.js

```
import React, { Component } from 'react';export default class
Parent extends Component {
render() {
    return (
      <div>
          <h1>I am parent Component</h1>
      </div>
    );
 }
}
```

Child.js

```
import React, { Component } from 'react';export default class Child
extends Component {
render() {
    return (
      <div>
          <h3>I am Child Component</h3>
      </div>
    );
 }
}
```

Now we will do the following tasks :

1. We will call the Child component inside Parent component
2. We will Pass some value from Parent component to Child component

1)Calling Child component inside Parent component:

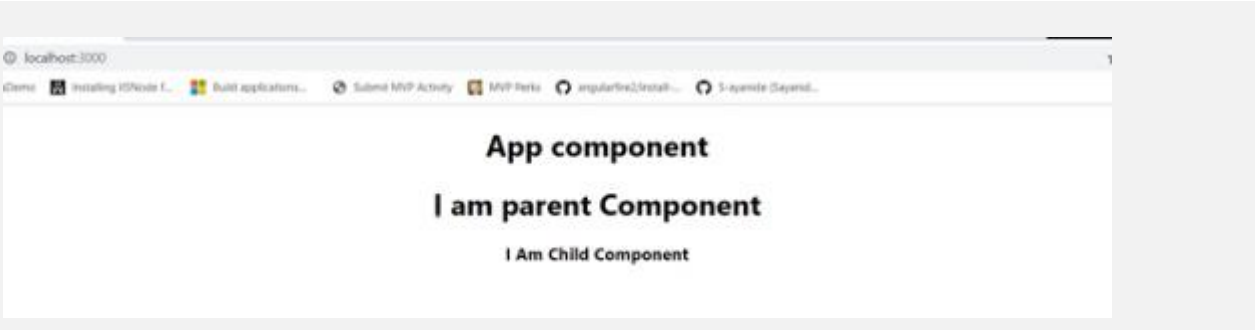
For this, we need to **import** this Child component on **Parent.js**

```
import Child from './Child';
```

And now just call this <child></child> inside render() method.

```
render() {
   return (
       <div>
            <h1>I am parent Component</h1>
            <Child></Child>
        </div>
   );
}
```

Now run this app and we can see the contents of both Parent and Child component.



2)Passing some value from Parent to Child component:

On **Parent.js**, we will declare a property as 'Title' for Child and assign some value to it.

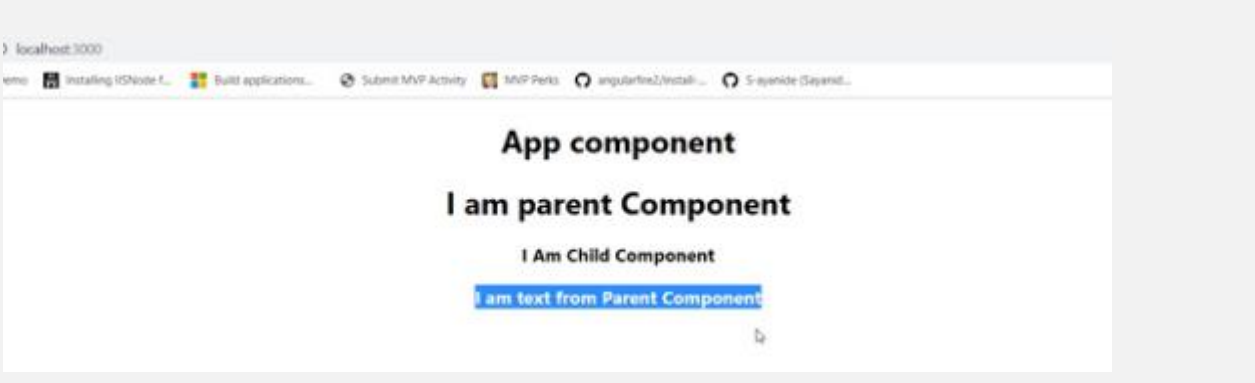
```
import React, { Component } from 'react';
import Child from './Child';export default class Parent extends
Component {
render() {
   return (
       <div>
             <h1>I am parent Component</h1>
             <Child Title="I am text from Parent
Component"></Child>
        </div>
   );
 }
}
```

And now on **Child.js**, we will simply fetch this '**Title**' set by our Parent component. This **Title** attribute is accessed using **this.props.Title**.

To do this we need to use JSX syntax and as you know for this we need to use { } to print the value.

```
import React, { Component } from 'react';export default class Child
extends Component {
render() {
    return (
      <div>
          <h3>I am Child Component</h3>
          <h3>{this.props.Title}</h3>
      </div>
    );
 }
}
```

Check the browser and you can see that "I am text from Parent Component", which was sent by Parent to Child component.



Using Props with Functional Components

On **Parent.js,** let us call our functional component **Demo1.js**.

Import and render the Demo1 component here as below:

```
import React, { Component } from 'react';
import Child from './Child';\
import Demo1 from './Demo1'export default class Parent extends
Component {
  render() {
      return (
        <div>
           <h1>I am parent Component</h1>
           <Child Title="I am text from Parent Component">
</Child>
           <Demo1 Title="I am Text for function Component"
></Demo1>
       </div>
   );
 }
}
```

You can see that we have set **Title** property with this Demo1 component too. And like the previous example, we will also get this **Title** on Demo1 (functional) component.

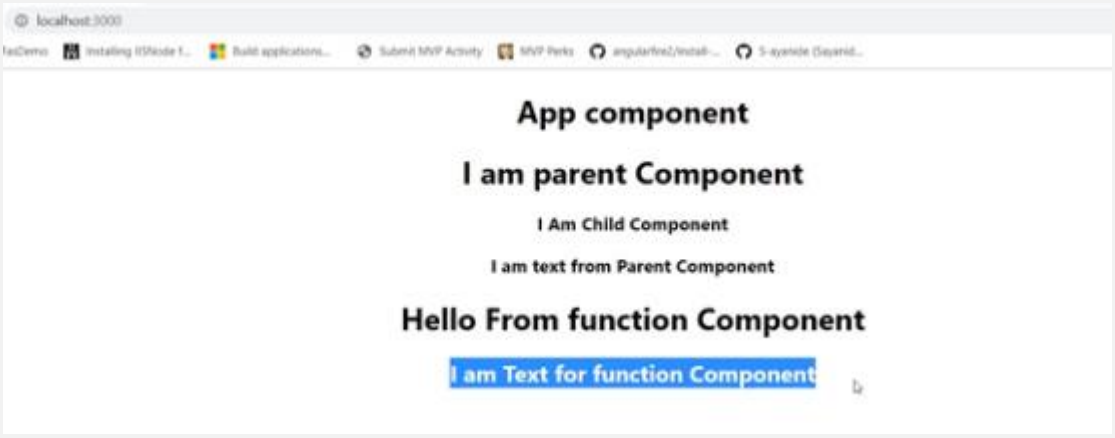
Remember that in the functional component we do not have **this** keyword to fetch the value of prop. So here we have pass **props** as an argument to our functional component. And then we can use **props. Title** to get the value.

Demo1.js

```
import React from 'react';const Demo1 = (props) => {
   return (
      <div>
          <h1>Hello From function Component</h1>
```

```
          <h2>{props.Title}</h2>
      </div>
   );
}export default Demo1;
```

And run and check the output.



What is State?

- The behavior of the app/component at a given moment in time is defined by the state.
- Components data will be stored in the component's state.
- This state can be modified based on user action or other action.
- When a component state is changed, React will re-render the component to the browser.

Let's understand **State** with an example:

We will create a class component and name it as
a **Sample** component.

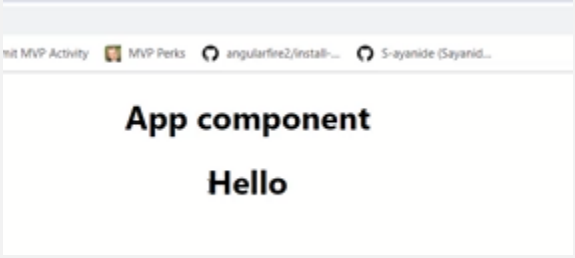
In this, firstly we will simply create a variable and then display its value. Then we will have a Button and by clicking on this Button we will change the value of this variable.

Sample.js

```
import React, { Component } from "react";export default class
Sample extends Component {
state = {
    a: 'Hello'
};render() {
   return (
      <div>
         <h1>{this.state.a}</h1>
      </div>
  );
 }
}
```

As you can see we have declared a variable inside state and to retrieve the state, we'll have **this.state.a** which uses the same ES6 method as before.

Now run it and check on the browser, you will see "Hello" is there.

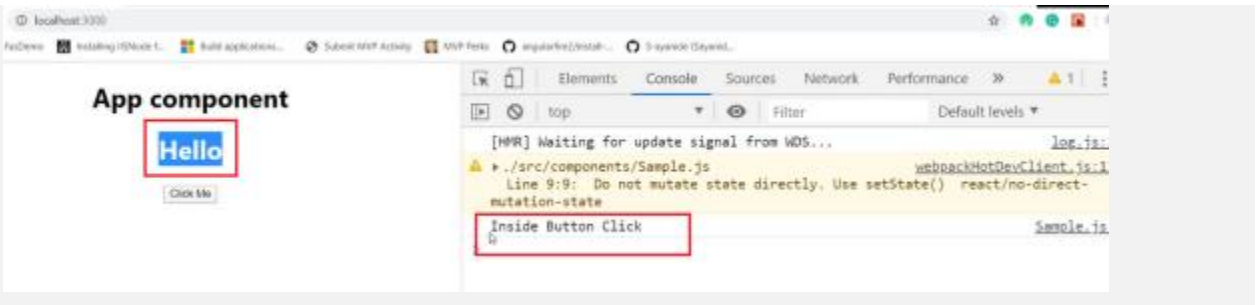


Now let us have a Button and we will have its onClick method. As you know by clicking this we will change the text from “Hello” to some other in our <h1> tag.

Sample.js

```
import React, { Component } from "react";export default class
Sample extends Component {
state = {
    a: 'Hello'
};handleButtonClick = () => {
   console.log("Inside Button Click");
   this.state.a = "You pressed Button";
};render() {
   return (
      <div>
         <h1>{this.state.a}</h1>
         <button type="button" onClick={this.handleButtonClick}
>Click Me</button>
      </div>
  );
 }
}
```

You can see that we have created **handleButtonClick()**method of the button and inside that event, we are changing the variable text with the state.



You have noticed that when you pressed that button the variable text is not updating but it is printing the console.!

So why does this happen?

For the solution read the warning that your console is showing.

It is saying that you can not mutate the state of a variable directly and you need to use **setState()** for this.

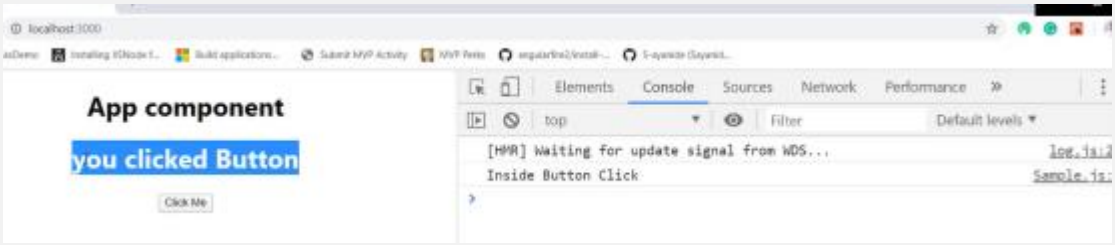
So to update the state, we'll use **this.setState( )**. This is a built-in method to modify the state.

Sample.js

```
import React, { Component } from "react";export default class
Sample extends Component {
state = {
    a: 'Hello'
};handleButtonClick = () => {
   console.log("Inside Button Click");
   // this.state.a = "You pressed Button";
   this.setState({
       a: "you clicked Button"
   });
};render() {
   return (
      <div>
```

```
          <h1>{this.state.a}</h1>
          <button type="button" onClick={this.handleButtonClick}
>Click Me</button>
       </div>
  );
 }
}
```

Now run the app and click that button:



So, when you clicked that button you got the changed text.

State Vs Props

- Props are immutable i.e. once we set the props then it cannot be changed, while State is an observable object that is used to hold data that may change over time and also used to control the behavior after each and every change.
- While Props are set by the parent component, State is generally updated by event handlers.