The main difference between state and props is that **props** are immutable. This is why the container component should define the state that can be updated and changed, while the child components should only pass data from the state using props.

Using Props

When we need immutable data in our component, we can just add props to **reactDOM.render()** function in **main.js** and use it inside our component.

App.jsx

import React from 'react';

class App extends React.Component {

render() {

return (

<div>

<h1>{this.props.headerProp}</h1>

<h2>{this.props.contentProp}</h2>

</div>

);

}

}

export default App;

main.js

import React from 'react';

import ReactDOM from 'react-dom';

import App from './App.jsx';

ReactDOM.render(<App headerProp = "Header from props..." contentProp = "Content

from props..."/>, document.getElementById('app'));

export default App;

This will produce the following result.



Default Props

You can also set default property values directly on the component constructor instead of adding it to the **reactDom.render()** element.

App.jsx

import React from 'react';

class App extends React.Component {

render() {

return (

<div>

<h1>{this.props.headerProp}</h1>

<h2>{this.props.contentProp}</h2>

</div>

);

}

}

App.defaultProps = {

headerProp: "Header from props...",

contentProp:"Content from props..."

}

export default App;

main.js

import React from 'react';

import ReactDOM from 'react-dom';

import App from './App.jsx';

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

Output is the same as before.



State and Props

The following example shows how to combine **state** and props in your app. We are setting the state in our parent component and passing it down the component tree using **props**. Inside the **render** function, we are setting **headerProp** and **contentProp** used in child components.

App.jsx

import React from 'react';

class App extends React.Component {

constructor(props) {

super(props);

this.state = {

header: "Header from props...",

content: "Content from props..."

}

}

render() {

return (

<div>

<Header headerProp = {this.state.header}/>

<Content contentProp = {this.state.content}/>

</div>

);

}

}

class Header extends React.Component {

render() {

return (

<div>

<h1>{this.props.headerProp}</h1>

</div>

);

}

}

class Content extends React.Component {

render() {

return (

<div>

<h2>{this.props.contentProp}</h2>

</div>

);

}

}

export default App;

main.js

import React from 'react';

import ReactDOM from 'react-dom';

import App from './App.jsx';

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

The result will again be the same as in the previous two examples, the only thing that is different is the source of our data, which is now originally coming from the **state**. When we want to update it, we just need to update the state, and all child components will be updated. More on this in the Events chapter.

