مروری بر Props مربوط به ReactJS

تفاوت اصلی بین state و props این است که prop ها تغییر ناپذیر هستند. دلیل این امر این است که جزء نگهدارنده باید حالتی را تعریف کند که بتوان آن را به روز رسانی کرده و تغییر داد. این در حالی است که اجزای فرزند تنها باید داده ها را از حالت با استفاده از props عبور دهند.

استفاده از props

اگر در جزء خود به داده های تغییرناپذیری نیاز داریم، می توانیم صرفا props را به تابع reactDOM.render() موجود در main.js اضافه کنیم و از آن داخل جزء خود استفاده کنیم.

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;
```

این کار باعث می شود نتیجه ی زیر نمایش داده شود.

Props پیش فرض

می توانید مقادیر مشخصه ی پیش فرض را مستقیما در سازنده ی جزء تنظیم کنید، به جای آن که آن را به عنصر reactDom.render() اضافه کنید.

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'));
```

خروجی مانند قبل است.

State و Props مربوط به ReactJS

در مثال زیر چگونگی ترکیب state و props در برنامه نشان داده شده است. ما state را در جزء مادر تنظیم کرده ایم و با استفاده از props آن را به درخت جزء عبور داده ایم. headerProp و contentProp استفاده شده در اجزای فرزند را داخل تابع render تنظیم کرده ایم.

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'));
```

نتیجه باز هم تغییری نخواهد کرد و تنها چیزی که تغییر می کند، منبع داده های ما است که در حال حاضر از state سرچشمه می گیرند. اگر بخواهیم این برنامه را به روز رسانی کنیم، باید state را به روز رسانی کنیم و به دنبال آن تمامی اجزای فرزند به روز رسانی خواهند شد. در بخش رویدادها بیشتر به این مطلب پرداخته شده است.