چرخه ی عمر اجزا(Component Life Cycle) در ReactJS

در این بخش می خواهیم به متدهای چرخه ی عمر اجزا بپردازیم.

متدهای چرخه ی عمر

- componentWillMountقبل از رندر شدن در هر دو سمت کلاینت و سرور اجرا می شود.
- componentDidMount پس از اولین رندر و تنها در سمت کلاینت اجرا می شود. در همین مکان است که درخواست های AJAX و DOM یا به روز رسانی حالات رخ می دهد. این متد در یکپارچه شدن با فریمورک های جاوا اسکریپت دیگر و هر تابعی که اجرای آن با تأخیر همراه است، مثل setTimeout و setInterval کاربرد دارد. ما برای به روز کردن حالت از این متد استفاده می کنیم تا بتوانیم متدهای چرخه ی عمر دیگر را فعال کنیم.
- componentWillReceiveProps به محض به روز شدن ویژگی ها قبل از فراخوانی رندر دیگر احضار می شود. ما این متد را از setNewNumber و زمانی که حالت را به روز کردیم، فعال کردیم.
- shouldComponentUpdate مقدار true یا false را برگشت می دهد. این متد به روز بودن یا نبودن جزء را مشخص می کند و به صورت پیش فرض بر روی true قرار دارد. اگر مطمئن هستید که جزء مورد نظر شما پس از به روز شدن حالت یا ویژگی ها نیازی به رندر ندارد، می توانید مقدار false را برگشت دهید.
- componentWillUpdate درست قبل از رندر شدن فراخوانی می شود.
- componentDidUpdate درست بعد از رندر شدن فراخوانی می شود.
- componentWillUnmount پس از جدا شدن جزء از DOM فراخوانی می شود. ما جزء خود را در main.js از DOM جدا می کنیم.

در مثال زیر حالت اولیه ی تابع سازنده را مشخص کرده ایم. جهت به روز رسانی حالت از setNewnumber استفاده شده است. تمامی متدهای چرخه ی عمر داخل جزء Content قرار دارند.

App.jsx

```
import React from 'react';


class App extends React.Component {
  constructor(props) {
    super(props);


    this.state = {
      data: 0
    }
    this.setNewNumber = this.setNewNumber.bind(this)
  };
  setNewNumber() {
    this.setState({data: this.state.data + 1})
  }
  render() {
    return (
      <div>
        <button onClick = {this.setNewNumber}>INCREMENT</button>
        <Content myNumber = {this.state.data}></Content>
      </div>
    );
  }
}
class Content extends React.Component {
  componentWillMount() {
```

```
  console.log('Component WILL MOUNT!')
}
componentDidMount() {
  console.log('Component DID MOUNT!')
}
componentWillReceiveProps(newProps) {
  console.log('Component WILL RECIEVE PROPS!')
}
shouldComponentUpdate(newProps, newState) {
  return true;
}
componentWillUpdate(nextProps, nextState) {
  console.log('Component WILL UPDATE!');
}
componentDidUpdate(prevProps, prevState) {
  console.log('Component DID UPDATE!')
}
componentWillUnmount() {
  console.log('Component WILL UNMOUNT!')
}
render() {
  return (
    <div>
      <h3>{this.props.myNumber}</h3>
    </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'));



setTimeout(() => {

  ReactDOM.unmountComponentAtNode(document.getElementById('app'));}, 10000);
```

پس از رندر اولیه نتیجه ی زیر نمایش داده می شود.