React Lifecycle. 43Rd/100 **Day**

Section 2

Yesterday we covered the mounting phase of react component lifecycle, today we are going to handle updating and tomorrow we will handle the unmounting phase, but make sure you clearly understand the mounting phase to make sense out of this part.

Updating.

After component mounting the next phase in the lifecycle is when a component is updated. A component is updated whenever there is a change in the component's state or props. React has five built–in methods that gets called, in this order, when a component is updated:

1). getDerivedStateFromProps()

2). shouldComponentUpdate()

3). render()

4). getSnapshotBeforeUpdate()

5). componentDidUpdate()

The render() method is required and will always be called, the others are optional and will be called if you define them.

1). getDerivedStateFromProps

Also at updates the getDerivedStateFromProps method is called. This is the first method that is called when a component gets updated.

This is still the natural place to set the state object based on the initial props.

The example below has a button that changes the favorite color to blue, but since the getDerivedStateFromProps() method is called, which updates the state with the color from the favcol attribute, the favorite color is still rendered as yellow:

Example:

```
class Header extends React.Component {
 constructor(props) {
    super(props);
    this.state = {favoritecolor: "red"};
  }
  static getDerivedStateFromProps(props, state) {
    return {favoritecolor: props.favcol };
  }
  changeColor = () => {
    this.setState({favoritecolor: "blue"});
  }
  render() {
    return (
      <div>
      <h1>My Favorite Color is {this.state.favoritecolor}</h1>
      <button type="button" onClick={this.changeColor}>Change
color</button>
      </div>
    );
  }
}

ReactDOM.render(<Header favcol="yellow"/>,
document.getElementById('root'));
```

2). **shouldComponentUpdate**

In the shouldComponentUpdate() method you can return a Boolean value that specifies whether React should continue with the rendering or not.

The default value is true.

Example below shows what happens when the shouldComponentUpdate() method returns false:

Example:

```
class Header extends React.Component {
 constructor(props) {
    super(props);
    this.state = {favoritecolor: "red"};
  }
  shouldComponentUpdate() {
    return false;
  }
  changeColor = () => {
    this.setState({favoritecolor: "blue"});
  }
  render() {
    return (
      <div>
      <h1>My Favorite Color is {this.state.favoritecolor}</h1>
      <button type="button" onClick={this.changeColor}>Change
color</button>
      </div>
    );
  }
}

ReactDOM.render(<Header />, document.getElementById('root'));
```

3). **render**

The render() method is of course called when a component gets updated, it has to re-render the HTML to the DOM, with the new changes.

The example below has a button that changes the favorite color to blue:

Example:

```
class Header extends React.Component {
 constructor(props) {
    super(props);
    this.state = {favoritecolor: "red"};
  }
  changeColor = () => {
    this.setState({favoritecolor: "blue"});
  }
  render() {
    return (
      <div>
      <h1>My Favorite Color is {this.state.favoritecolor}</h1>
      <button type="button" onClick={this.changeColor}>Change
color</button>
      </div>
    );
  }
}

ReactDOM.render(<Header />, document.getElementById('root'));
```

4). **getSnapshotBeforeUpdate**

In the getSnapshotBeforeUpdate() method you have access to the props and state before the update, meaning that even after the update, you can check what the values were before the update.

If the getSnapshotBeforeUpdate() method is present, you should also include the componentDidUpdate() method, otherwise you will get an error.

The example below might seem complicated, but all it does is this:

When the component is mounting it is rendered with the favorite color "red".

When the component has been mounted, a timer changes the state, and after one second, the favorite color becomes "yellow".

This action triggers the update phase, and since this component has a getSnapshotBeforeUpdate() method, this method is executed, and writes a message to the empty DIV1 element.

Then the componentDidUpdate() method is executed and writes a message in the empty DIV2 element:(Next Page)

Example:

```
class Header extends React.Component {
  constructor(props) {
    super(props);
    this.state = {favoritecolor: "red"};
  }
  componentDidMount() {
    setTimeout(() => {
      this.setState({favoritecolor: "yellow"})
    }, 1000)
  }
  getSnapshotBeforeUpdate(prevProps, prevState) {
    document.getElementById("div1").innerHTML =
    "Before the update, the favorite was " + prevState.favoritecolor;
  }
  componentDidUpdate() {
    document.getElementById("div2").innerHTML =
    "The updated favorite is " + this.state.favoritecolor;
  }
  render() {
    return (
      <div>
        <h1>My Favorite Color is {this.state.favoritecolor}</h1>
        <div id="div1"></div>
        <div id="div2"></div>
      </div>
    );
  }
}

ReactDOM.render(<Header />, document.getElementById('root'));
```

5). **componentDidUpdate**

The componentDidUpdate method is called after the component is updated in the DOM.

The example below might seem complicated, but all it does is this:

When the component is mounting it is rendered with the favorite color "red". When the component has been mounted, a timer changes the state, and the color becomes "yellow". This action triggers the update phase, and since this component has a componentDidUpdate method, this method is executed and writes a message in the empty DIV element:

Example:

```
class Header extends React.Component {
  constructor(props) {
    super(props);
    this.state = {favoritecolor: "red"};
  }
  componentDidMount() {
    setTimeout(() => {
      this.setState({favoritecolor: "yellow"})
    }, 1000)
  }
  componentDidUpdate() {
    document.getElementById("mydiv").innerHTML =
    "The updated favorite is " + this.state.favoritecolor;
  }
  render() {
    return (
      <div>
      <h1>My Favorite Color is {this.state.favoritecolor}</h1>
      <div id="mydiv"></div>
      </div>
    );}}
ReactDOM.render(<Header />, document.getElementById('root'))
```