8

React Life Cycle

Objectives

After completing this lesson, you should be able to do the following:

- Component Creation
- Component Removal
- Component Update
- Lifecycle method call in different states

React Component Lifecycle

- Lifecycle methods are to be used to run code and interact with your component at different points in the components life.
- These methods are based around a component Mounting, Updating, and Unmounting.

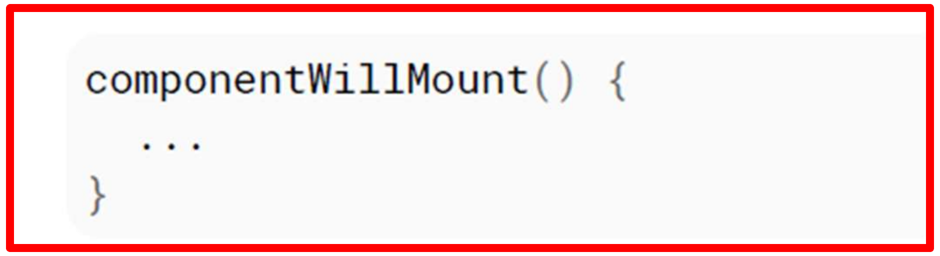
Component Creation

- When a React component is created, a number of functions are called:
- If you are using class Component extends React.Component (ES6), 3 user defined functions are called.

1. componentWillMount()

This is the **First** method called.

- This function can be used to make final changes to the component before it will be added to the DOM.

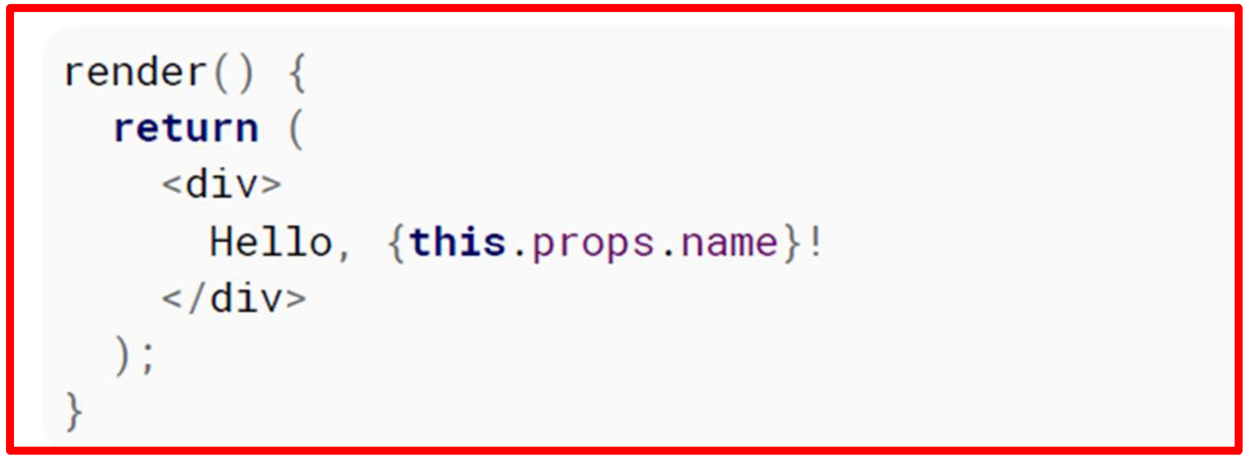


```
componentWillMount() {
  ...
}
```

2. render()

This is the **Second** method called.

- The render() function should be a pure function of the component's state and props. It returns a single element which represents the component during the rendering process and should either be a representation of a native DOM component (e.g. **<p />**) or a composite component.
- If nothing should be rendered, it can return **null** or **undefined.**
- This function will be recalled after any change to the component's props or state.

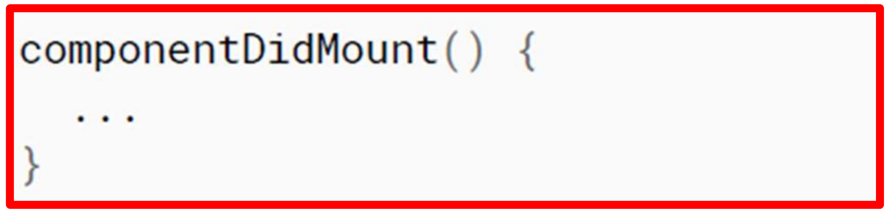


```
render() {
  return (
    <div>
      Hello, {this.props.name}!
    </div>
  );
}
```

3. componentDidMount()

This is the **Third** method called.

- The component has been mounted and you are now able to access the component's DOM nodes, e.g. via refs.
- This method should be used for:
  1. Preparing timers
  2. Fetching data
  3. Adding event listeners
  4. Manipulating DOM elements



```
componentDidMount() {
  ...
}
```

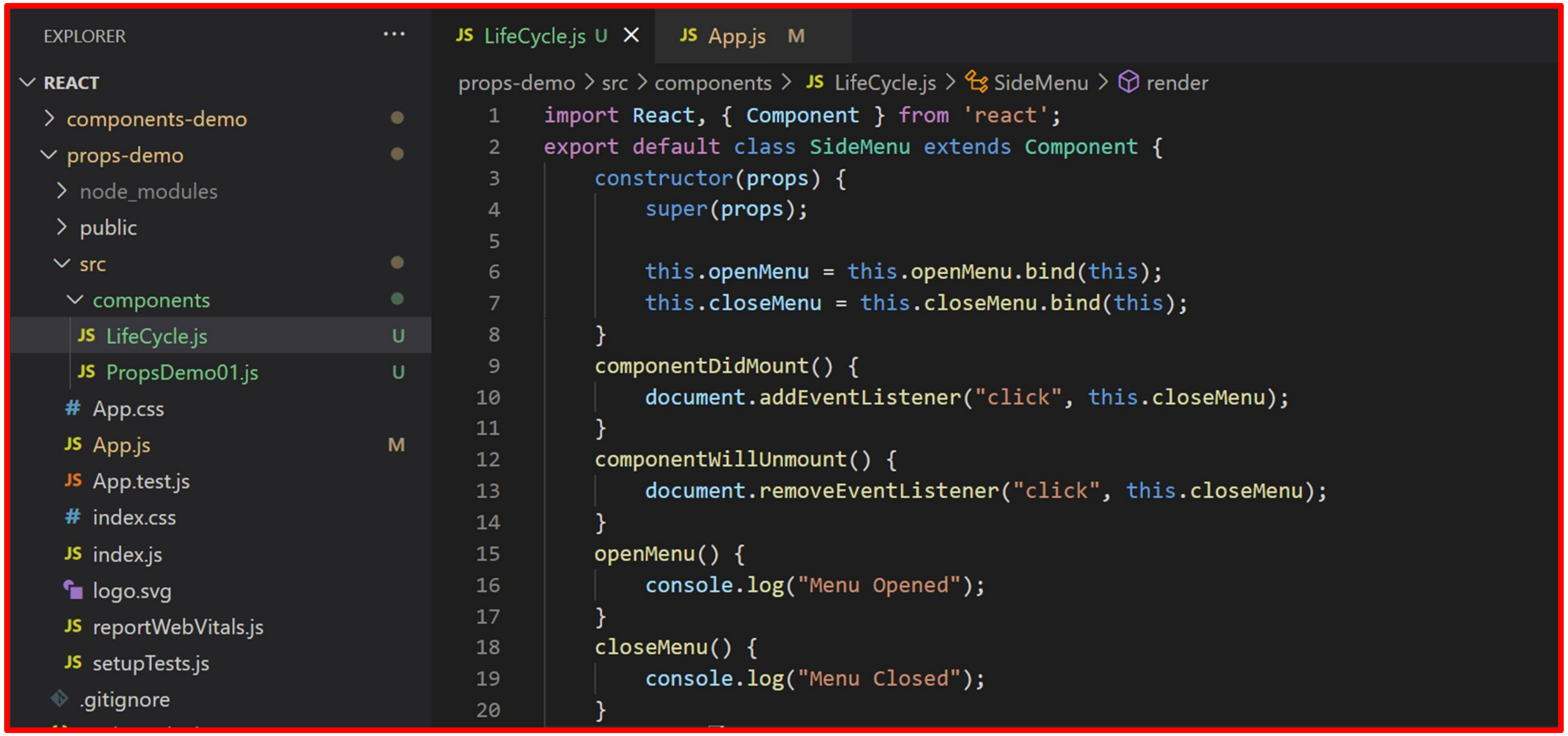
Component Removal

componentWillUnmount()

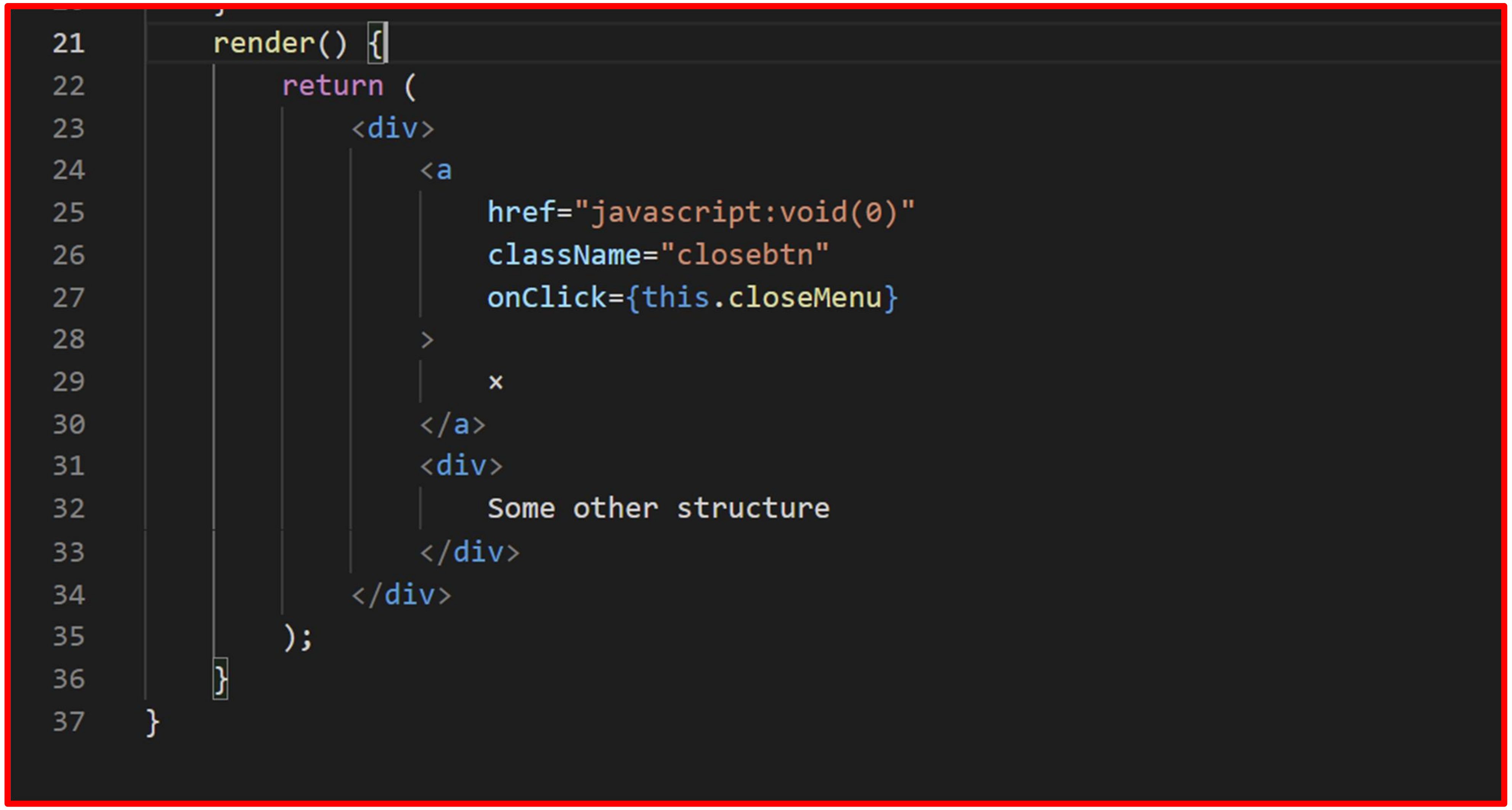
- This method is called **before** a component is unmounted from the DOM.
- It is a good place to perform cleaning operations like:
  1. Removing event listeners.
  2. Clearing timers.
  3. Stopping sockets.
  4. Cleaning up redux states.

```
componentWillUnmount(){
  ...
}
```

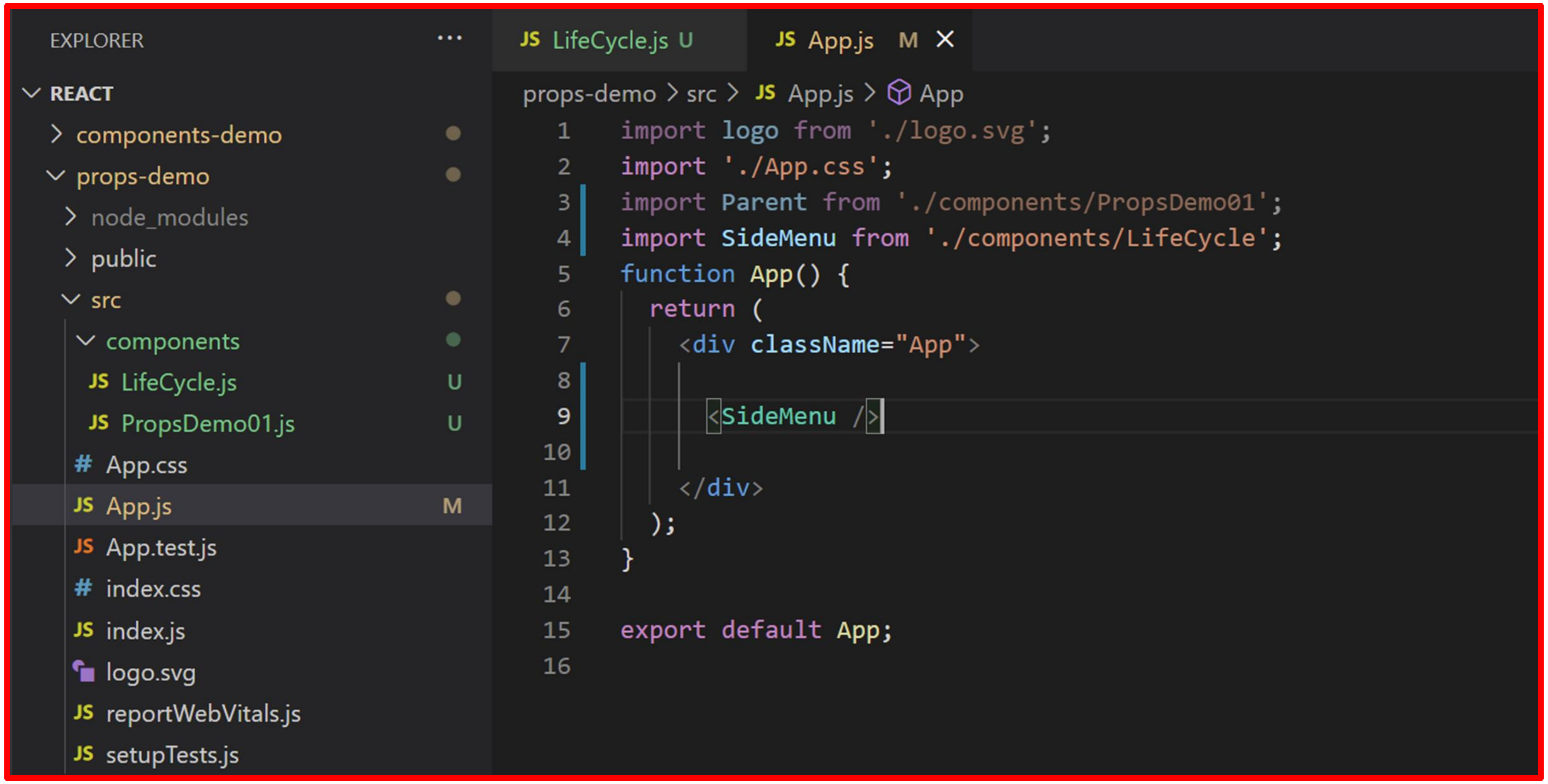
An example of removing attached event listener in componentWillUnMount



```
import React, { Component } from 'react';
export default class SideMenu extends Component {
    constructor(props) {
        super(props);

        this.openMenu = this.openMenu.bind(this);
        this.closeMenu = this.closeMenu.bind(this);
    }
    componentDidMount() {
        document.addEventListener("click", this.closeMenu);
    }
    componentWillUnmount() {
        document.removeEventListener("click", this.closeMenu);
    }
    openMenu() {
        console.log("Menu Opened");
    }
    closeMenu() {
        console.log("Menu Closed");
    }
```



```
    render() {
        return (
            <div>
                <a
                    href="javascript:void(0)"
                    className="closebtn"
                    onClick={this.closeMenu}
                >
                    ×
                </a>
                <div>
                    Some other structure
                </div>
            </div>
        );
    }
}
```

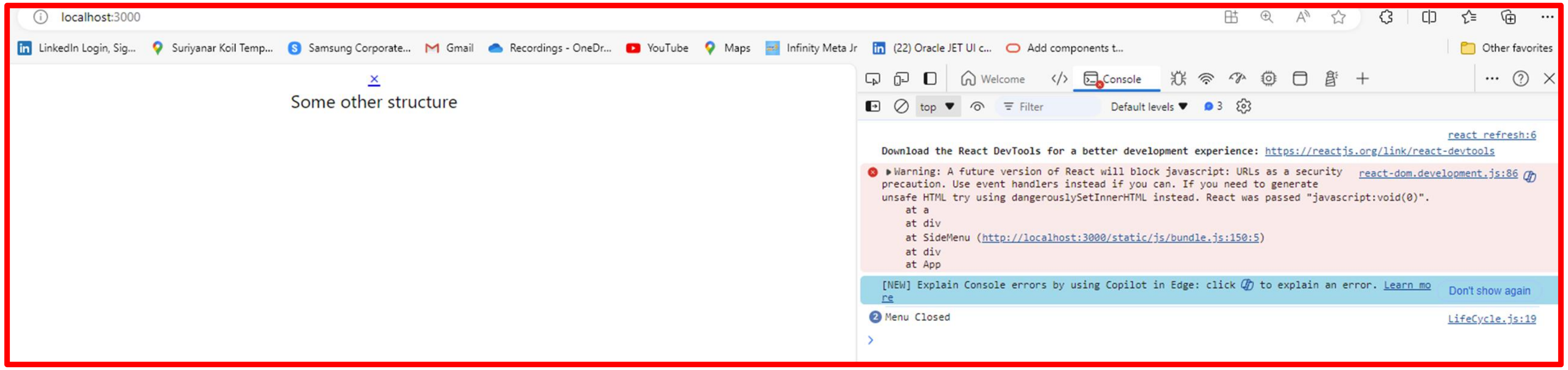


EXPLORER ···

REACT
- components-demo
- props-demo
  - node_modules
  - public
  - src
    - components
      - LifeCycle.js U
      - PropsDemo01.js U
    - App.css
    - App.js M
    - App.test.js
    - index.css
    - index.js
    - logo.svg
    - reportWebVitals.js
    - setupTests.js

```
import logo from './logo.svg';
import './App.css';
import Parent from './components/PropsDemo01';
import SideMenu from './components/LifeCycle';
function App() {
  return (
    <div className="App">

      <SideMenu />

    </div>
  );
}

export default App;

```

Output



X
Some other structure

```
                                                                          react refresh:6
Download the React DevTools for a better development experience: https://reactjs.org/link/react-devtools
▸Warning: A future version of React will block javascript: URLs as a security    react-dom.development.js:86
precaution. Use event handlers instead if you can. If you need to generate
unsafe HTML try using dangerouslySetInnerHTML instead. React was passed "javascript:void(0)".
    at a
    at div
    at SideMenu (http://localhost:3000/static/js/bundle.js:150:5)
    at div
    at App
[NEW] Explain Console errors by using Copilot in Edge: click  to explain an error. Learn mo
re
2 Menu Closed                                                             LifeCycle.js:19
>
```

Component Update

`componentWillReceiveProps(nextProps)`

This is the **first function called on properties changes**.

When **component's properties change**, React will call this function with the **new properties**. You can access to the old props with *this.props* and to the new props with *nextProps*.

With these variables, you can do some comparison operations between old and new props, or call function because a property change, etc.

```
componentWillReceiveProps(nextProps){
  if (nextProps.initialCount && nextProps.initialCount > this.state.count){
    this.setState({
      count : nextProps.initialCount
    });
```

`shouldComponentUpdate(nextProps, nextState)`

This is the **second function called on properties changes and the first on state changes**.

By default, if another component / your component change a property / a state of your component, **React** will render a new version of your component. In this case, this function always return true.

You can override this function and **choose more precisely if your component must update or not**.

This function is mostly used for **optimization**.

In case of the function returns **false**, the **update pipeline stops immediately**.

```
componentShouldUpdate(nextProps, nextState){
  return this.props.name !== nextProps.name ||
    this.state.count !== nextState.count;
}
```

`componentWillUpdate(nextProps, nextState)`

This function works like `componentWillMount()`. **Changes aren't in DOM**, so you can do some changes just before the update will perform.

```
componentWillUpdate(nextProps, nextState){}
```

`render()`

There's some changes, so re-render the component.

`componentDidUpdate(prevProps, prevState)`

Same stuff as `componentDidMount()` : **DOM is refreshed**, so you can do some work on the DOM here.

```
componentDidUpdate(prevProps, prevState){}
```

Lifecycle method call in different states

When a component is initialized:

1. componentWillMount
2. render
3. componentDidMount

When a component has state changed:

1. shouldComponentUpdate
2. componentWillUpdate
3. render
4. componentDidUpdate

When a component has props changed:

1. componentWillReceiveProps
2. shouldComponentUpdate
3. componentWillUpdate
4. render
5. componentDidUpdate

When a component is unmounting:

1. componentWillUnmount

Summary

In this lesson, you should have learned how to:

- Component Creation
- Component Removal
- Component Update
- Lifecycle method call in different states